MAS
Modula–2 Algebra System

Specifications
Definition Modules
Indexes

Computer Algebra Group
University of Passau

MAS Version 1.0[1]

[1]Document revision October 1, 1996

Copyrights

Preface

This document contains summarys and indexes which are useful for working with the MAS system. The organization of this document is as follows:

Chapter 1 gives an short introduction in using the different informations containted in this document.

Chapter 2 contains the specifications of several algebraic structures.

Chapters 3 to 12 list all Modula-2 definition modules of the ALDES /SAC-2, the DIP and the MAS libraries.

Chapter 13 consists of an procedure header to definition module index.

Chapter 14 contains a algorithm comment by keyword to procedure name index.

The last part is an procedure name index of this document.

Acknowledgements

Many thanks to all who made contributions or influenced this project: R. Loos, G. E. Collins and co–workers for the ALDES / SAC–2 system; I. Giese, W. Kynast and H. Czytrek for discussions in the early stages of the MAS project; M. Pesch, B. Haible, V. Weispfenning and T. Becker for feedback during the later stages of the development of MAS. In version 0.7 contributions from the following students are incorporated: B. Deyle, H. Friebel, E. Reisinger, M. Rothmeier, K. Rieger, T. Wollersberger, J. Philipp, J. Müller, E. Schönfeld, T. Belkahia, W. Mark. In the current version contributions from the students M. Göbel, F. Lippold, C. Rose, J. Pfeil, A. Dolzmann, R. Grosse-Gehling, I. Bader have been included.

The MAS system was first developed using an Atari 1040 ST and is now available on most Unix worksations, DOS and OS/2 PCs.

Passau, October 1, 1996. H. Kredel[1], M. Pesch[2]

[1]University of Mannheim, L 15, 16, D-68131 Mannheim, FRG, Tel: (49,0) 621/ 292 5673, E-mail: kredel @ rz.uni-mannheim.de

[2]University of Passau, Innstraße 33, D-94030 Passau, FRG, Tel: (49,0) 851/ 509-3123, E-mail: pesch @ alice.fmi.uni-passau.de

Contents

Chapter 1

Overview and Introduction

This document contains useful information for programming and computing with MAS.

In the first sections we discuss the structure of the information contained in the following chapters. In the last section we provide some hints on how to retrieve the information from this document.

1.1 Specifications

The chapter on specifications contains some listings of first test structures. These specifications may change in later versions.

The listings have been printed using some LaTeX macros distributed by Eamann McManus. The macros combine two character tokens into one TeX math–operator. The transliteration is given in table 1.1.

1.1.1 Overview

The first section in this chapter contains an initialization data set, which reads all required specifications of algebraic structures and exposes them in the top level environment.

The second section defines some abstract structures like Abelian Groups (`AGROUP`) or Fields (`FIELD`).

The third section contains the definition of all built–in structures such as lists (`Lists`), booleans (`BOOL`) or atoms (`ATOM`).

The fourth section contains the specifications of the arithmetic structures such as integers (`INTEGER`), rational numbers (`RATIONAL`), modular integers (`MODINT`) or arbitrary precision floating point numbers (`FLOAT`).

The fifth section defines the polynomial system structures such as integral recursive polynomials (`IPOL`), distributive rational polynomials (`DIRP`), sets (lists) of distributive rational polynomials (`DIRL`) and rings with Gröbner bases (`GBRING`).

The sixth section contains the definition of some term models such as propositional logic (`PROPLOG`) or Peano arithmetic (`PEANO`).

Modula–2 string	printed TEX character
!=	$\neq$
<<	$\ll$
<=	$\leq$
<>	$\neq$
>>	$\gg$
>=	$\geq$
->	$\rightarrow$
:=	$\leftarrow$
^	$\uparrow$
"character string"	"character␣string"
'character string'	'character␣string'

Table 1.1: Transliteration in Listings

The last section contains some test examples for the evaluation of expression using the specifications.

1.2 Libraries

The library chapters contain the Modula–2 definition modules of MAS. Not contained are probably defined global variables of the modules.

The comments on the procedures are structured according the ALDES / SAC–2 publication scheme. The first sentence of the comment defines the full procedure identification. The procedure name is a mnemonic abreviation of this full identification. The remaining text describes the input and output parameters of the procedure together with a short description of its purpose and functionality.

The comments are extracted from the implementation ALDES source files so they may differ in some cases to the official publication ALDES versions. Only the destinction between capital letters and lower case letters has been introduced. The ornamentations are still in implementation ALDES style. However the identifier names in the Modula–2 implementation modules still correspond to the implementation ALDES names. Some care is needed in case of the function return parameters. They are no more visible in the Modula–2 procedure header, but should be determinable from the comment text without difficulty.

The procedure names are put to the master index at the end of the document.

Information on the data structure definitions is partly contained in the *MAS Interactive Usage* document or directly in the Modula–2 implementation modules.

Some statistics on system source code sizes are summarized in table 1.2. The indicated numbers refer to the *distributed* systems without further packages which may be available somewhere.

System	lines	1000 bytes
MAS 1.0	101103	3 395
MAS 0.7	67005	2 238
MAS 0.6	42023	1 484
ALDES/SAC-2	15791	1 279
ALDES-2/SAC-2	19398	1 571
saclib 1.1	34519	783
Macaulay	19521	389
GAP 3.1, library	70585	2 287
GAP 3.1, C kernel	46681	1 765
SiMATH 3.5	175520	3 949
Reduce 3.5	100100	3 432
Reduce 3.4	117080	3 997
Reduce 3.3	51661	1 665
Scratchpad II	66382	2 402
Axiom	93054	3 451
Mathematica	38424	1 290
Maple V3 (maple.lib)	?	(10 083)
Maple V2 (maple.lib)	?	(5 620)

Table 1.2: System Source Code Sizes

1.3 Indexes

There are three indexes:

1. a procedure header to definition module index,
2. a comment keyword to procedure name index and
3. a master index of procedure names.

In the sequel we discuss the definition of each index.
In case of the procedure header to definition module index each line is build as follows:

1. The first column shows the name of the Modula–2 definition module. The file name extension `.DEF` is not shown.
2. The second column shows the Modula–2 (ALDES / SAC–2) procedure name.
3. The rest of the line shows the input and output parameters of the procedures.
4. The index is alphabeticaly sorted by the second column.

In case of the comment keyword to procedure name index the first line gives the keyword, then each following line up to the next keyword is build as follows:

1. The first column shows the name of the Modula–2 (ALDES / SAC–2) procedure.

2. The second column shows the name identifying comment sentence.
3. The index is alphabeticaly sorted by the keywords.

Finally the master index of procedure names contains all names of procedures listed in the definition modules. The numbers following the procedure name show the page number in this document where the procedure comment is listed.

1.4 How to Use the Indexes

There are three main usages of this document

1. you know the procedure name and you want to know the procedure parameters or the procedure location.
2. you have some idea of a procedure function and want to know if there exists a procedure which implements this function.
3. you know the procedure name and want to know the description of the functionality of the procedure.

The first problem can be solved in two ways:

1. You search the procedure name in the procedure header to definition module index. From the found entry the number of parameters and (in most cases) from the parameter names the types of the parameters can be determined.
2. Or you search the procedure name in the master index. From the found entry the page number of the procedure comment can be determined. Looking at this page the description of the procedure parameters together with the functionality of the procedure can be determined.

For the second problem there are also two solutions:

1. You search for a procedure comment in the comment to procedure name index. From the found entry the names of the desired procedures can be determined.
2. You search in the table of contents for the definition module with the required functionality. Then from the listing of the definition module the existance of an appropriate procedure can be determined.

For the third problem there is only one recomendable solution:

1. You search the procedure name in the master index. From the found entry the page number of the procedure comment can be determined. Looking at this page the description of the procedure parameters together with the functionality of the procedure can be determined.

Note that for the interactive use of MAS it is required that you check if the procedure is accessible from the interpreter by the `help(name,Loaded)` or `help(all)` or `HELP` or `EXTPROCS` command.

Chapter 2

Specifications

The description of the specification component is contained in chapter 4 of the "MAS Interactive Usage" Manual. In this chapter we list some example specifications together with some application examples.

2.1 Initialization

```
BEGIN
 CLOUT("Loading␣specifications␣...");
 (*OUT("NUL:");*) x←T; (* define T for parser *)
 PRAGMA(SLOPPY); (*PRAGMA(DEBUG); PRAGMA(TRACE);*)
 (*PRAGMA(GENPARSE);  ensure non-generic parse *)
 END.

(* read required specifications. *)

IN("spec\abstr.spc").   (* abstract items *)

IN("spec\lisp.spc").    (* builtin lisp functions *)

IN("spec\arith.spc").   (* basic arithmetic *)

IN("spec\term.spc").    (* elementary term structures *)

IN("spec\modul.spc").   (* free module *)

IN("spec\poly.spc").    (* polynomials *)


CLOUT("␣...␣exposing␣units␣...␣").
(*PRAGMA(DEBUG). PRAGMA(TRACE).*)
```

```
(*specifics. *)
EXPOSE ATOM.
EXPOSE REPRESENTATION.
EXPOSE Lists.

EXPOSE INTEGER.
EXPOSE RATIONAL.
EXPOSE MODINT.
EXPOSE FLOATING.

EXPOSE PROPLOG.
EXPOSE PEANO.

EXPOSE IPOL.
EXPOSE DIRP.
EXPOSE DIRL.

(*generics. *)
EXPOSE OBJECT.
EXPOSE AGROUP.
EXPOSE RING.
EXPOSE FIELD.
EXPOSE GBRING.

EXPOSE MODULE(RING).

EXPOSE FMODULE(INTEGER,INT,INT).

BEGIN
 PRAGMA(FUSSY); (*PRAGMA(DEBUG); PRAGMA(TRACE);*)
 (*SHUT("NUL:");*)
 (*PRAGMA(GENPARSE); allow generic parse, if required *)
 CLOUT("...␣finished.");
 END.
```

2.2 Abstract Structures

```
(* ———- specifications of abstract items — *)

SPECIFICATION OBJECT;
(*Object specification. *)
(*1*) SORT obj;
(*2*) SIGNATURE READ     (obj) : obj;
      SIGNATURE WRITE    (obj) ;
(*3*) SIGNATURE DECREAD  (obj) : obj;
      SIGNATURE DECWRITE (obj) ;
```

```
(*4*) SIGNATURE DEFAULT  (obj) : obj;
      SIGNATURE COERCE   (obj) : obj;
(*9*) END OBJECT.


SPECIFICATION AMONO;
(*Abelian monoid specification. *)
(*1*) IMPORT OBJECT[ amono/obj ];
(*2*) SIGNATURE ZERO (amono)       : amono;
(*3*) SIGNATURE SUM  (amono,amono) : amono;
(*9*) END AMONO.


SPECIFICATION AGROUP;
(*Abelian group specification. *)
(*1*) IMPORT AMONO[ ag/amono ];
(*2*) SIGNATURE DIF  (ag,ag) : ag;
      SIGNATURE NEG  (ag)    : ag;
(*9*) END AGROUP.


SPECIFICATION XRING;
(*Ring specification extending A group. *)
(*1*) IMPORT AGROUP[ xring/ag ];
(*2*) SIGNATURE ONE  (xring)       : xring;
(*3*) SIGNATURE PROD (xring,xring) : xring;
      SIGNATURE EXP  (xring,xring) : xring;
(*9*) END XRING.


SPECIFICATION RING;
(*Ring specification combining an A group and an A moniod. *)
(*1*) SORT atom;
      IMPORT AGROUP[ ring/ag ];
      IMPORT AMONO [ ring/amono, ONE/ZERO, PROD/SUM ];
(*3*) SIGNATURE EXP  (ring,atom) : ring;
(*9*) END RING.


SPECIFICATION MODULE(Ring (*: spec*));
(*Module specification. *)
(*1*) IMPORT AGROUP[ melem/ag ];
      (*IMPORT Ring;*)
      SORT ring;
(*3*) SIGNATURE SPROD (ring,melem) : melem;
(*9*) END MODULE.
```

```
SPECIFICATION ERING;
(*Euclidean ring specification. *)
(*1*) IMPORT RING[ er/ring ];
(*2*) SIGNATURE QR  (er,er,er,er) ;
      SIGNATURE GCD (er,er)       : er;
(*9*) END ERING.


SPECIFICATION GBRING;
(*Groebner base ring specification. *)
      IMPORT RING[ gbring/ring ];
(*2*) SIGNATURE NF  (gbring,ring) : ring;
      SIGNATURE GB  (gbring)      : gbring;
      SIGNATURE IRR (gbring)      : gbring;
(*9*) END GBRING.


SPECIFICATION XFIELD;
(*Field specification extending ring. *)
(*1*) IMPORT RING[ xfield/ring ];
(*2*) SIGNATURE REZIP (xfield)        : xfield;
      SIGNATURE Q     (xfield,xfield) : xfield;
(*9*) END XFIELD.


SPECIFICATION FIELD;
(*Field specification combining two A groups. *)
(*1*) SORT atom;
      IMPORT AGROUP[ field/ag ];
      IMPORT AGROUP[ field/ag, ONE/ZERO, PROD/SUM, REZIP/NEG, Q/DIF ];
(*3*) SIGNATURE EXP  (field ,atom) : field;
(*9*) END FIELD.


(* ------ implementations of abstract items -- *)

IMPLEMENTATION RING;
(*1*) PROCEDURE EXP(X,n);
      VAR   x: ring;
      VAR   i: atom;
      BEGIN
      (*1*) IF n ≤ 0 THEN x←ONE(X); RETURN(x) END;
      (*3*) i←n; x←X;
            WHILE i > 1 DO i←i−1;
                  x←PROD(x,X) END;
            RETURN(x)
      (*9*) END EXP;
(*9*) END RING.
```

```
(*
AXIOMS FIELD;
(*1*) RULE EXP(X,n) => exp(X,n)          WHEN n ≥ 0;
      RULE EXP(X,n) => exp(REZIP(X),−n) WHEN n < 0;
(*9*) END FIELD.
*)
```

2.3 LISP Structures

```
(* ———- atom unit — *)

SPECIFICATION ATOM;
(*Atoms specification. *)
(*1*) SORT atom;
(*2*) SIGNATURE Aone     ()          : atom;
      SIGNATURE Azero    ()          : atom;
(*3*) SIGNATURE MUL      (atom,atom) : atom;
      SIGNATURE ADD      (atom,atom) : atom;
      SIGNATURE SUB      (atom)      : atom;
      SIGNATURE SUB      (atom,atom) : atom;
      SIGNATURE POW      (atom,atom) : atom;
      SIGNATURE QUOT     (atom,atom) : atom;
      SIGNATURE REM      (atom,atom) : atom;
(*4*) SIGNATURE Adefault ()          : atom;
(*9*) END ATOM.


IMPLEMENTATION ATOM;
(*1*) PROCEDURE Aone();
      BEGIN RETURN(1) END Aone;
(*2*) PROCEDURE Azero();
      BEGIN RETURN(0) END Azero;
(*3*) PROCEDURE Adefault();
      BEGIN RETURN(0) END Adefault;
(*9*) END ATOM.


MODEL RING;
(*Atoms are a model for rings. *)
(*1*) IMPORT ATOM;
      IMPORT INTEGER;
(*2*) MAP READ(atom)      → IREAD();
      MAP WRITE(atom)     → IWRITE(VAL);
```

```
     MAP DEFAULT(atom)    → Adefault();
(*3*) MAP ONE(atom)       → Aone();
     MAP ZERO(atom)       → Azero();
(*4*) MAP PROD(atom,atom) → MUL(VAL,VAL);
     MAP SUM(atom,atom)   → ADD(VAL,VAL);
     MAP DIF(atom,atom)   → SUB(VAL,VAL);
     MAP NEG(atom)        → SUB(VAL);
(*9*) END RING.


(* ———- list unit — *)

SPECIFICATION Lists; (*LIST is a keyword ! *)
(*List processing specification. *)
(*1*) SORT list, object, atom;
(*2*) SIGNATURE NullList ()                    : list;
     SIGNATURE COMP     (object, list)         : list;
     SIGNATURE LIST     (object, (*...,*) object) : list;
     SIGNATURE CONC     (list,list)            : list;
     SIGNATURE CCONC    (list,list)            : list;
(*3*) SIGNATURE FIRST   (list)                 : object;
     SIGNATURE RED      (list)                 : list;
     SIGNATURE ADV      (list, object, list)   ;
(*4*) SIGNATURE INV     (list)                 : list;
     SIGNATURE CINV     (list)                 : list;
(*5*) SIGNATURE SFIRST  (list,object)          ;
     SIGNATURE SRED     (list,list)            ;
(*7*) SIGNATURE EQUAL   (object,object)        : atom;
     SIGNATURE LENGTH   (list)                 : atom;
(*9*) END Lists.


IMPLEMENTATION Lists;
(*1*) PROCEDURE NullList(): list;
     RETURN(LIST()) NullList;
(*9*) END Lists.


(* ———- bool unit — *)

SPECIFICATION BOOL;
(*Boolean specification. *)
(*1*) SORT bool, object;
(*2*) SIGNATURE true     ()             : bool;
     SIGNATURE false    ()             : bool;
(*3*) SIGNATURE EQ       (object,object) : bool;
     SIGNATURE NE       (object,object) : bool;
     SIGNATURE LE       (object,object) : bool;
```

```
    SIGNATURE GE      (object,object) : bool;
    SIGNATURE LT      (object,object) : bool;
    SIGNATURE GT      (object,object) : bool;
(*4  (*key words*)
    SIGNATURE OR      (bool,bool)     : bool;
    SIGNATURE AND     (bool,bool)     : bool;
    SIGNATURE NOT     (bool)          : bool;
    *)
(*9*) END BOOL.


IMPLEMENTATION BOOL;
(*1*) PROCEDURE true();
    BEGIN RETURN(T) END true;
(*2*) PROCEDURE false();
    BEGIN RETURN(LIST()) END false;
(*9*) END BOOL.


(* ——— representation unit — *)

SPECIFICATION REPRESENTATION;
(*Representation specification. *)
(*1*) SORT elem, name, rep, pair, bool, func;
(*2*) SIGNATURE NewRep      ()              : rep;
    SIGNATURE SetRep      (name,elem,rep) ;
    SIGNATURE GetRep      (name,rep)      : elem;
    SIGNATURE CopyRep     (rep)           : rep;
    SIGNATURE StepRep     (rep)           : pair;
    SIGNATURE ForEachinRep (rep,func)     : rep;
    SIGNATURE FullRep     (rep)           : bool;
    SIGNATURE Pair        (name,elem)     : pair;
    SIGNATURE NullPair    ()              : pair;
(*9*) END REPRESENTATION.


IMPLEMENTATION REPRESENTATION;
(*1*) PROCEDURE Pair(a,b): pair;
    RETURN(LIST(a,b)) Pair;
(*2*) PROCEDURE NullPair(): pair;
    RETURN(LIST()) NullPair;
(*9*) END REPRESENTATION.
```

2.4 Arithmetic Structures

```
(* ———- integer unit — *)

SPECIFICATION INTEGER;
(*Integral numbers specification. *)
(*1*) SORT INT;
(*2*) SIGNATURE IWRITE (INT)          ;
      SIGNATURE IREAD  (INT)          : INT;
(*3*) SIGNATURE Ione   ()             : INT;
      SIGNATURE Izero  ()             : INT;
(*4*) SIGNATURE IPROD  (INT,INT)      : INT;
      SIGNATURE ISUM   (INT,INT)      : INT;
      SIGNATURE IDIF   (INT,INT)      : INT;
      SIGNATURE INEG   (INT)          : INT;
      SIGNATURE IQR    (INT,INT,INT,INT) ;
      SIGNATURE IGCD   (INT,INT)      : INT;
(*9*) END INTEGER.


IMPLEMENTATION INTEGER;
(*1*) PROCEDURE Ione();
      BEGIN RETURN(1) END Ione;
(*2*) PROCEDURE Izero();
      BEGIN RETURN(0) END Izero;
(*9*) END INTEGER.


MODEL RING;
(*Integers are a model for rings. *)
(*1*) IMPORT INTEGER;
(*2*) MAP READ(INT)      → IREAD();
      MAP WRITE(INT)     → IWRITE(VAL);
(*3*) MAP ONE(INT)       → Ione();
      MAP ZERO(INT)      → Izero();
(*4*) MAP PROD(INT,INT)  → IPROD(VAL,VAL);
      MAP SUM(INT,INT)   → ISUM(VAL,VAL);
      MAP DIF(INT,INT)   → IDIF(VAL,VAL);
      MAP NEG(INT)       → INEG(VAL);
(*9*) END RING.


(* ———- rational number unit — *)

SPECIFICATION RATIONAL;
(*Rational numbers specification. *)
(*1*) SORT RAT, INT, atom;
(*2*) SIGNATURE RNWRITE (RAT)     ;
      SIGNATURE RNDRD   (RAT)     : RAT;
(*3*) SIGNATURE RNone   ()        : RAT;
```

```
    SIGNATURE RNzero  ()        : RAT;
(*4*) SIGNATURE RNPROD  (RAT,RAT) : RAT;
    SIGNATURE RNSUM   (RAT,RAT) : RAT;
    SIGNATURE RNDIF   (RAT,RAT) : RAT;
    SIGNATURE RNNEG   (RAT)     : RAT;
    SIGNATURE RNINV   (RAT)     : RAT;
    SIGNATURE RNQ     (RAT,RAT) : RAT;
(*5*) SIGNATURE RNINT   (INT)     : RAT;
    SIGNATURE RNprec  (atom)    ;
(*9*) END RATIONAL.


IMPLEMENTATION RATIONAL;
    VAR s: atom;
(*1*) PROCEDURE RNone();
    BEGIN RETURN(RNINT(1)) END RNone;
(*2*) PROCEDURE RNzero();
    BEGIN RETURN(RNINT(0)) END RNzero;
(*3*) PROCEDURE RNWRITE(a);
    BEGIN IF s < 0 THEN RNWRIT(a) ELSE RNDWR(a,s) END;
        END RNWRITE;
(*4*) PROCEDURE RNprec(a);
    BEGIN s←a END RNprec;
(*8*) BEGIN
        s←−1;
(*9*) END RATIONAL.


MODEL FIELD;
(*Rational numbers are a model for fields. *)
(*1*) IMPORT RATIONAL;
(*2*) MAP READ(RAT)      → RNDRD();
    MAP WRITE(RAT)     → RNWRITE(VAL);
(*3*) MAP ONE(RAT)       → RNone();
    MAP ZERO(RAT)      → RNzero();
(*4*) MAP PROD(RAT,RAT)  → RNPROD(VAL,VAL);
    MAP SUM(RAT,RAT)   → RNSUM(VAL,VAL);
    MAP DIF(RAT,RAT)   → RNDIF(VAL,VAL);
    MAP NEG(RAT)       → RNNEG(VAL);
    MAP Q(RAT,RAT)     → RNQ(VAL,VAL);
    MAP REZIP(RAT)     → RNINV(VAL);
(*9*) END FIELD.


(* ———- modular integer unit — *)

SPECIFICATION MODINT;
(*Modular integers specification. *)
```

```
(*1*) SORT MI, INT, mod;
(*2*) SIGNATURE MIWRIT (MI)         ;
      SIGNATURE MIREAD (mod,MI)     : MI;
(*3*) SIGNATURE MIone  (mod)        : MI;
      SIGNATURE MIzero ()           : MI;
(*4*) SIGNATURE MIPROD (mod,MI,MI)  : MI;
      SIGNATURE MISUM  (mod,MI,MI)  : MI;
      SIGNATURE MIDIF  (mod,MI,MI)  : MI;
      SIGNATURE MINEG  (mod,MI)     : MI;
      SIGNATURE MIINV  (mod,MI)     : MI;
      SIGNATURE MIQ    (mod,MI,MI)  : MI;
(*5*) SIGNATURE MIHOM  (mod,INT)    : MI;
(*9*) END MODINT.


IMPLEMENTATION MODINT;
(*1*) PROCEDURE MIone(m);
      BEGIN RETURN(MIHOM(m,1)) END MIone;
(*2*) PROCEDURE MIzero();
      BEGIN RETURN(0) END MIzero;
(*3*) PROCEDURE MIREAD(m);
      BEGIN RETURN(MIHOM(m,IREAD())) END MIREAD;
(*4*) PROCEDURE MIWRIT(x);
      BEGIN IWRITE(x) END MIWRIT;
(*9*) END MODINT.


MODEL FIELD;
(*Integers mod p are a model for fields. *)
(*1*) IMPORT MODINT;
      IMPORT INTEGER;
(*2*) MAP READ(MI)      → MIREAD(DESC);
      MAP WRITE(MI)     → MIWRIT(VAL);
      MAP DECREAD(MI)   → IREAD();
      MAP DECWRITE(MI)  → IWRITE(VAL);
(*3*) MAP ONE(MI)       → MIone(DESC);
      MAP ZERO(MI)      → MIzero();
(*4*) MAP PROD(MI,MI)   → MIPROD(DESC,VAL,VAL) WHEN EQ(DESC,DESC);
      MAP SUM(MI,MI)    → MISUM(DESC,VAL,VAL)  WHEN EQ(DESC,DESC);
      MAP DIF(MI,MI)    → MIDIF(DESC,VAL,VAL)  WHEN EQ(DESC,DESC);
      MAP NEG(MI)       → MINEG(DESC,VAL);
      MAP Q(MI,MI)      → MIQ(DESC,VAL,VAL)    WHEN EQ(DESC,DESC);
      MAP REZIP(MI)     → MIINV(DESC,VAL);
(*9*) END FIELD.


(* ———- floating point number unit — *)
```

```
SPECIFICATION FLOATING;
(*Floating point numbers specification. *)
(*1*) SORT FLOAT, INT, RAT, atom;
(*2*) SIGNATURE APWRIT (FLOAT)       ;
      SIGNATURE APREAD (FLOAT)       : FLOAT;
(*3*) SIGNATURE APone  ()            : FLOAT;
      SIGNATURE APzero ()            : FLOAT;
(*2*) SIGNATURE APPROD (FLOAT,FLOAT) : FLOAT;
      SIGNATURE APSUM  (FLOAT,FLOAT) : FLOAT;
      SIGNATURE APDIF  (FLOAT,FLOAT) : FLOAT;
      SIGNATURE APNEG  (FLOAT)       : FLOAT;
      SIGNATURE APINV  (FLOAT)       : FLOAT;
      SIGNATURE APQ    (FLOAT,FLOAT) : FLOAT;
(*5*) SIGNATURE APFINT (INT)         : FLOAT;
      SIGNATURE APFRN  (RAT)         : FLOAT;
      SIGNATURE RNFAP  (FLOAT)       : RAT;
      SIGNATURE APSPRE (atom)        ;
(*9*) END FLOATING.


IMPLEMENTATION FLOATING;
(*1*) PROCEDURE APone();
      BEGIN RETURN(APFINT(1)) END APone;
(*2*) PROCEDURE APzero();
      BEGIN RETURN(APFINT(0)) END APzero;
(*3*) PROCEDURE APDIF(a,b);
      BEGIN RETURN(APDIFF(a,b)) END APDIF;
(*4*) PROCEDURE APINV(a);
      BEGIN RETURN(APQ(APFINT(1),a)) END APINV;
(*9*) END FLOATING.


MODEL FIELD;
(*AP floating numbers are (nearly) a model for fields. *)
(*1*) IMPORT FLOATING;
(*2*) MAP READ(FLOAT)        → APREAD();
      MAP WRITE(FLOAT)       → APWRIT(VAL);
(*3*) MAP ONE(FLOAT)         → APone();
      MAP ZERO(FLOAT)        → APzero();
(*4*) MAP PROD(FLOAT,FLOAT)  → APPROD(VAL,VAL);
      MAP SUM(FLOAT,FLOAT)   → APSUM(VAL,VAL);
      MAP DIF(FLOAT,FLOAT)   → APDIF(VAL,VAL);
      MAP NEG(FLOAT)         → APNEG(VAL);
      MAP Q(FLOAT,FLOAT)     → APQ(VAL,VAL);
      MAP REZIP(FLOAT)       → APINV(VAL);
(*9*) END FIELD.
```

2.5 Polynomial Structures

```
(* ———- integer recursive polynomial unit — *)

SPECIFICATION IPOL;
(*Integer recursive polynomial specification. *)
(*1*) SORT ipol, pol, dip, atom;
(*2*) SIGNATURE IPzero ()              : ipol;
      SIGNATURE IPSUM  (atom,ipol,ipol)  : ipol;
      SIGNATURE IPNEG  (atom,ipol)       : ipol;
      SIGNATURE IPDIF  (atom,ipol,ipol)  : ipol;
(*3*) SIGNATURE IPPROD (atom,ipol,ipol)  : ipol;
      SIGNATURE IPEXP  (atom,ipol,ipol)  : ipol;
(*4*) SIGNATURE PFDIP  (dip,atom,pol)    ;
(*9*) END IPOL.


IMPLEMENTATION IPOL;
(*2*) PROCEDURE IPzero();
      BEGIN RETURN(0) END IPzero;
(*2*) PROCEDURE IPDIF(r,a,b);
      BEGIN RETURN(IPSUM(r,a,IPNEG(r,b))) END IPDIF;
(*9*) END IPOL.


MODEL RING;
(*Integer recursive polynomial are a model for rings. *)
(*1*) IMPORT IPOL;
(*2*) MAP ZERO(ipol)      → IPzero();
      MAP SUM(ipol,ipol)  → IPSUM(DESC,VAL,VAL)  WHEN EQ(DESC,DESC);
      MAP NEG(ipol)       → IPNEG(DESC,VAL);
      MAP DIF(ipol,ipol)  → IPDIF(DESC,VAL,VAL)  WHEN EQ(DESC,DESC);
(*3*) MAP PROD(ipol,ipol) → IPPROD(DESC,VAL,VAL) WHEN EQ(DESC,DESC);
      MAP EXP(ipol,atom)  → IPEXP(DESC,VAL,VAL);
(*9*) END RING.


(* ———- distributive rational polynomial unit — *)

SPECIFICATION DIRP;
(*Distributive rational polynomial specification. *)
(*1*) SORT dip, dirp, diip, pol, atom;
(*2*) SIGNATURE DIRPSM (dirp,dirp)        : dirp;
      SIGNATURE DIRPNG (dirp)            : dirp;
      SIGNATURE DIRPDF (dirp,dirp)       : dirp;
(*3*) SIGNATURE DIRPPR (dirp,dirp)       : dirp;
```

```
(*4*) SIGNATURE DIPFP  (atom,pol)        : dip;
(*6*) SIGNATURE DIIFRP (dirp)            : diip;
      SIGNATURE DIRFIP (diip)            : dirp;
(*9*) END DIRP.


IMPLEMENTATION DIRP;
(*2*) PROCEDURE DIRPDF(a,b);
      BEGIN RETURN(DIRPSM(a,DIRPNG(b))) END DIRPDF;
(*9*) END DIRP.


MODEL RING;
(*Distributive rational polynomial are a model for rings. *)
(*1*) IMPORT DIRP;
(*2*) MAP SUM(dirp,dirp)  → DIRPSM(VAL,VAL);
      MAP NEG(dirp)       → DIRPNG(VAL);
      MAP DIF(dirp,dirp)  → DIRPDF(VAL,VAL);
(*4*) MAP PROD(dirp,dirp) → DIRPPR(VAL,VAL);
      MAP EXP(dirp,dirp)  → DIRPEX(VAL,VAL);
(*9*) END RING.


(* ———- distributive rational polynomial list unit — *)

SPECIFICATION DIRL;
(*Distributive rational polynomial list specification. *)
(*1*) SORT dip, dirp, diip, pol, atom, dc;
(*2*) SIGNATURE PDREAD  ()          : dc;
      SIGNATURE PDWRITE (dc)        ;
(*3*) SIGNATURE DIRLRD  ()          : dirl;
      SIGNATURE DIRLWR  (dirl)      ;
      SIGNATURE PREADD  (dc)        : dirl;
      SIGNATURE PWRITED (dirl)      ;
(*4*) SIGNATURE DIILFR  (dirl)      : diil;
(*5*) SIGNATURE DIRPNF  (dirl,dirp) : dirp;
      SIGNATURE DIRPGB  (dirl,atom) : dirl;
      SIGNATURE DIRPGB1 (dirl)      : dirl;
      SIGNATURE DIRLIS  (dirl)      : dirl;
(*6*) SIGNATURE SetTflag(atom)      ;
(*9*) END DIRL.


IMPLEMENTATION DIRL;
(*Distributive rational polynomial list implementation. *)
      VAR s: atom;
(*2*) PROCEDURE DIRPGB1(a) : dirl;
      BEGIN RETURN(DIRPGB(a,s)) END DIRPGB1;
```

```
(*3*) PROCEDURE SetTflag(a);
      BEGIN s←a END SetTflag;
(*4*) BEGIN
          s←1;
(*9*) END DIRL.
```

```
MODEL OBJECT;
(*Distributive rational polynomial lists are a model for objects. *)
(*1*) IMPORT DIRL;
(*2*) MAP DECREAD(dirl)   → PDREAD();
      MAP DECWRITE(dirl)  → PDWRITE(VAL);
(*3*) MAP READ(dirl)      → PREADD(DESC);
      MAP WRITE(dirl)     → PWRITED(DESC,VAL);
(*9*) END OBJECT.
```

```
(* ———- gb structure unit — *)

MODEL GBRING;
(*Distributive rational polynomial lists are a model for
Groebner base rings. *)
      IMPORT DIRL;
(*2*) MAP NF(dirl,dirp) → DIRPNF(VAL,VAL);
      MAP GB(dirl)      → DIRPGB1(VAL);
      MAP IRR(dirl)     → DIRLIS(VAL);
(*9*) END GBRING.
```

2.6 Term Structures

```
(* ———- propositional logic unit — *)

SPECIFICATION PROPLOG;
(*Propositional logic specification. *)
(*1*) SORT bool;
(*2*) SIGNATURE FALSE ()            : bool;
      SIGNATURE TRUE  ()            : bool;
(*3*) SIGNATURE and (bool,bool)     : bool;
      SIGNATURE or  (bool,bool)     : bool;
      SIGNATURE not (bool)          : bool;
      SIGNATURE implies (bool,bool) : bool;
      SIGNATURE equival (bool,bool) : bool;
      SIGNATURE test    (bool,bool) : bool;
(*9*) END PROPLOG.
```

```
AXIOMS PROPLOG;
(*Axioms for propositional logic. *)
(*1*) RULE not(TRUE())          => FALSE();
      RULE not(FALSE())         => TRUE();
(*2*) RULE and(TRUE(),TRUE())   => TRUE();
      RULE and(FALSE(),TRUE())  => FALSE();
      RULE and(TRUE(),FALSE())  => FALSE();
      RULE and(FALSE(),FALSE()) => FALSE();
(*3*) RULE or(TRUE(),TRUE())    => TRUE();
      RULE or(FALSE(),TRUE())   => TRUE();
      RULE or(TRUE(),FALSE())   => TRUE();
      RULE or(FALSE(),FALSE())  => FALSE();
(*4*) RULE implies(X,Y)         => or(not(X),Y);
      RULE equival(X,Y)         => and(implies(X,Y),implies(Y,X));
      RULE test(X,Y)            => X WHEN EQUAL(X,Y) = 0;
(*9*) END PROPLOG.


(* ———- peano arithmetic unit — *)

SPECIFICATION PEANO;
(*Peano structure specification. *)
(*1*) SORT nat;
      IMPORT PROPLOG;
(*2*) SIGNATURE null  ()        : nat;
      SIGNATURE one   ()        : nat;
    (* SIGNATURE succ  (nat)     : nat; *)
      SIGNATURE add   (nat,nat) : nat;
      SIGNATURE prod  (nat,nat) : nat;
(*3*) SIGNATURE equal (nat,nat) : bool;
    (* SIGNATURE pred  (nat)     : bool; *)
(*9*) END PEANO.


AXIOMS PEANO;
(*Axioms for Peano system. *)
      RULE equal(X,X)               => TRUE();
      RULE equal(succ(X),null())    => FALSE();
      RULE equal(null(),succ(X))    => FALSE();
(*1*) RULE equal(succ(null()),null()) => FALSE();
(*2*) RULE equal(succ(X),succ(Y))   => equal(X,Y);
(*3*) RULE add(X,null())            => X;
(*4*) RULE add(X,succ(Y))           => succ(add(X,Y));
(*5*) RULE prod(X,null())           => null();
(*6*) RULE prod(X,succ(Y))          => add(prod(X,Y),X);
(*7   RULE pred(Y)                  =>
            and(pred(null()),implies(pred(X),pred(succ(X)))); *)
```

```
    (*additional equations "lemmas" *)
    RULE add(succ(X),Y)          => succ(add(X,Y));
    RULE add(null(),X)           => X;
    RULE prod(succ(X),Y)         => add(X,prod(X,Y));
    RULE prod(null(),X)          => null();
    RULE one()                   => succ(null());
(*9*) END PEANO.


MODEL RING;
(*Peano numbers are a model for rings, except subtract. *)
(*1*) IMPORT PEANO;
(*2*) MAP WRITE(nat)    → MWRITE(VAL);
(*3*) MAP ONE(nat)      → one();
    MAP ZERO(nat)       → null();
(*4*) MAP PROD(nat,nat) → prod(VAL,VAL);
    MAP SUM(nat,nat)    → add(VAL,VAL);
(*9*) END RING.
```

2.7 Test Examples Input

```
CLOUT("Testing␣specification␣component␣...").

PRAGMA(GENPARSE).


(*Arithmetic units. *)

VAR i, j : INT.

i←"11111111111111111111111111111111111111111111111111111111111111".

j←i↑2.

j←j↑0−i*i+j.

j←j+ "333333333333333333": INT −j.



VAR r, s: RAT.

r←"2222222222.777777777777777".

s←r/r.
```

```
s←r↑0+s− "1": RAT.

VAR m, n: MI "7".
m←"11111111111111111111111111".
n←m/m.
n←m↑0+n− "8": MI "7".

VAR f, g: FLOAT.
f←"2222.3E10".
g←f↑3.
g←f/g.
g←g*f*f.

(*Term units. *)
x←one().
x←add(x,x).
y←prod(x,x).

PRAGMA(GENPARSE).
CLOUT("...finished.").
```

2.8 Test Examples Output

```
Testing specification component ...

ANS: INT.
ANS: "11111111111111111111111111111111111111111111111111111111111111".
```

Time: read = 0, eval = 33, print = 17, gc = 0.

ANS: `"12345679012345679012345679012345679012345679012345654320987654"`
`"320987654320987654320987654320987654320987654321"`.

Time: read = 17, eval = 16, print = 34, gc = 0.

ANS: `"1"`.

Time: read = 16, eval = 50, print = 0, gc = 0.

ANS: `"333333333333333333"`.

Time: read = 17, eval = 50, print = 0, gc = 0.

ANS: RAT.

ANS: `"2222222222777777777777777/1000000000000000"`.

Time: read = 0, eval = 16, print = 17, gc = 0.

ANS: `"1"`.

Time: read = 0, eval = 33, print = 0, gc = 0.

ANS: `"1"`.

Time: read = 17, eval = 67, print = 0, gc = 0.

ANS: MI.

Time: read = 16, eval = 17, print = 0, gc = 0.

ANS: `"4"`.

Time: read = 0, eval = 16, print = 17, gc = 0.

ANS: `"1"`.

Time: read = 17, eval = 16, print = 0, gc = 0.

ANS: `"1"`.

Time: read = 17, eval = 83, print = 0, gc = 0.

ANS: FLOAT.

Time: read = 17, eval = 0, print = 0, gc = 0.

ANS: "0.22223000000000000000000000000000000000000000000000000E14".

Time: read = 0, eval = 16, print = 100, gc = 0.

ANS: "0.109750892035670000000000000000000000000000000000000E41".

Time: read = 0, eval = 34, print = 66, gc = 0.

ANS: "0.20248582574415277276931900102751229788044580389018 1E-26".

Time: read = 0, eval = 17, print = 100, gc = 0.

ANS: "0.10000000000000000000000000000000000000000000000000E1".

Time: read = 0, eval = 50, print = 16, gc = 0.

ANS: succ(null()).

Time: read = 17, eval = 0, print = 0, gc = 0.

ANS: succ(succ(null())).

Time: read = 0, eval = 50, print = 16, gc = 0.

ANS: succ(succ(succ(succ(null())))).

Time: read = 0, eval = 167, print = 0, gc = 0.

MAS: ...finished.

Chapter 3

Kernel Algoritms

3.1 List Tools

procedure CLISTFA (atom:LIST):LIST;
{character list from atom. The decimal printable representation of atom is returned as a character list. }

procedure LIST6 (a1,a2,a3,a4,a5,a6:LIST):LIST;
{list of 6 elements. The list (a1,a2,...a6) is returned. }

procedure LPAIRS (L:LIST):LIST;
{list pairs. L=(l1,...ln) is a list. A list containing all lists (li,lj), where li and lj are elements of L and i ¡¿ j is returned. }

procedure LSRCHQ (a,L:LIST):LIST;
{List search equal. a is an element, L is a list. The first position of a in L is returned if a is member of L, otherwise 0 is returned. }

procedure UPCASE (clist:LIST):LIST;
{upcase character list. clist is a character list. All letters in clist are converted to upper case. The result is returned. }

3.2 MAS Basic I/O System

procedure BKSP ();
{Backspace. Reread the last character from the input stream. }

procedure BLINES (N : GAMMAINT);
{Blank lines. N is a positive integer. N records of one blank each are output. }

procedure CREAD (): GAMMAINT;
{Character read. Returns next character from the input stream. }

procedure CREADB (): GAMMAINT;
{Character read, skipping blanks. Returns next character from the input stream. }

procedure CWRITE (C : GAMMAINT);
{Character write. The character c is transmitted to the output stream. }

procedure CloseBIOS ();
{Close BIOS. Close all streams and write summary. }

procedure CUNIT (S : ARRAY OF CHAR): GAMMAINT;
{Close unit. The unit S is closed, with S as the external name. CUNIT returns 0 on successful completion, ne 0 else.}

procedure DIBUFF ();
{Display input buffer. The input buffer status is displayed.}

procedure DIGIT (C : GAMMAINT): BOOLEAN;
{Digit. c is a character. If c is a digit then TRUE is returned otherwise FALSE is returned. }

procedure GREAD (): GAMMAINT;
{Gamma-integer read. A gamma-integer is read from the input stream. any preceding blanks are skipped. }

procedure GWRITE (a : GAMMAINT);
{Gamma-integer write. The gamma-integer a is written in the output stream.}

procedure LETTER (C : GAMMAINT): BOOLEAN;
{Letter. c is a character. If c is a letter then TRUE is returned otherwise FALSE is returned. }

procedure LISTS (S : ARRAY OF CHAR): LIST;
{List from string. S is a character string with respect to local character code. A list if the corresponding ALDES character codes is returned.}

procedure SLIST (A : LIST; VAR S : ARRAY OF CHAR);
{String from list. A is a list of ALDES character codes. S is a the corresponding character string with respect to local character codes. }

procedure MASCHR (C : GAMMAINT): CHAR;
{MAS character. Returns the local character for the aldes character c. }

procedure MASORD (C : CHAR): GAMMAINT;
{MAS order. Returns the aldes code for the character c. }

procedure MASORDI (C : GAMMAINT): GAMMAINT;
{MAS order integer. Returns the aldes code for the integer c.}

procedure SILINE (VAR S, L, R : GAMMAINT);
{Set input line. The input line length is set to S, the left margin is set to L and the right margin is set to R. If any of the values of S, L or R is negative, then the corresponding value is left unchanged. The values in effect are returned. }

procedure SIUNIT (S : ARRAY OF CHAR): GAMMAINT;
{Set input unit. iunit is set to n, with s as the external name. siunit returns 0 on successful completion, ne 0 else.}

procedure SOLINE (VAR S, L, R : GAMMAINT);
{Set output line. The output line length is set to S, the left margin is set to L and the right margin is set to R. If any of the values of S, L or R is negative, then the corresponding value is left unchanged. The values in effect are returned. }

procedure SOUNIT (S : ARRAY OF CHAR): GAMMAINT;
{Set output unit. ounit is set to n, with s as the external name. sounit returns 0 on successful completion, ne 0 else. the current output buffer is emptied.}

procedure Summary ();
{Summary of stream IO. }

procedure StorSummary ();
{MASSTOR Summary. }

procedure SWRITE (S : ARRAY OF CHAR);
{String write. S is a character string with respect to local character codes. The single characters are converted to ALDES codes and written to the output stream. }

procedure TAB (n : GAMMAINT);
{Tabulate. n is a positive integer. if lmarg le n le rmarg then blanks are inserted in obuff until opos eq n.}

procedure IStreamKind (): INTEGER;
{Input stream kind. The kind of the current input stream is returned. }

procedure OStreamKind (): INTEGER;
{Output stream kind. The kind of the current output stream is returned. }

procedure EStreamKind (): INTEGER;
{Error stream kind. The kind of the current error stream is returned. }

3.3 MAS BIOS Utility

procedure INP (A: LIST): LIST;
{Input. Set input unit to stream A, A is an SAC-2 character list. }

procedure OUT (A: LIST): LIST;
{Output. Set output unit to stream A, A is an SAC-2 character list. }

procedure SHUT (A: LIST): LIST;
{Shut. Close stream A, A is an SAC-2 character list. }

procedure EDIT (A: LIST): LIST;
{Edit. Call editor for file A, A is an SAC-2 character list. }

procedure DOS (A: LIST): LIST;
{DOS. Call DOS program with parameters. A is an SAC-2 character list. }

procedure CLTIS (A: LIST);
{Character list to input stream. A is an SAC-2 character list. }

3.4 MAS Configuration

3.5 MAS Elementary Functions

procedure MASABS (a: GAMMAINT): GAMMAINT;
{Absolute value. a is a gamma-integer. Returns the absolute value of a. }

procedure MASEVEN (a: GAMMAINT): BOOLEAN;
{Even. a is a gamma-integer. Returns TRUE if a is even and FALSE otherwise. }

procedure MASEXP (a,b: GAMMAINT): GAMMAINT;
{Exponential function. a and b are gamma-integers, b non-negative. Returns $a\hat{}b$, with $0\hat{}0 = 1$. }

procedure MASMAX (a,b: GAMMAINT): GAMMAINT;
{Maximum. a and b are gamma-integers. Returns the maximum of a and b. }

procedure MASMIN (a,b: GAMMAINT): GAMMAINT;
{Minimum. a and b are gamma-integers. Returns the minimum of a and b. }

procedure MASODD (a: GAMMAINT): BOOLEAN;
{Odd. a is a gamma-integer. Returns TRUE if a is odd and FALSE otherwise. }

procedure MASQREM (a,b: GAMMAINT; VAR q,r: GAMMAINT);
{Quotient and remainder. a and b are gamma-integers, b non-zero. $q = integer(a/b)$ and $r = a - b * q$. }

procedure MASREM (a,b: GAMMAINT): GAMMAINT;
{Remainder. a and b are gamma-integers, b non-zero. Returns $a - b * integer(a/b)$. }

procedure MASSIGN (a: GAMMAINT): GAMMAINT;
{Sign. a is a gamma-integer. Returns the sign of a. }

3.6 MAS Error

procedure ERROR (a: GAMMAINT; s: ARRAY OF CHAR);
{Error. An error of severity a and indication s is reported. }

procedure ErrorHandler (a: P0): GAMMAINT;
{Error handler. Any error reported by the ERROR procedure is catched. }

3.7 MAS Signal Handling

procedure SigUsr1HandleDefault (signo: INTEGER);
{SIGUSR1 default signal handler. }

3.8 MAS Storage

procedure ADV (L: LIST; VAR a, LP: LIST);
{Advance. L is a non-null list. a=FIRST(L) and LP=RED(L). }

procedure CELLS (): GAMMAINT;
{Cells. Returns the used cells since storage initialization. }

procedure CLOCK (): GAMMAINT;
{Clock. Returns the current CPU clock reading in seconds. }

procedure COMP (a,L: LIST): LIST;
{Composition. a is an object. L is a list. Returns the composition of a and L. }

procedure DEQUE (L: LIST): LIST;
{Dequeue. L is a non empty queue representing list. Returns a, the first object from the queue. L is updated. }

procedure EMPTYQUE (M: LIST): BOOLEAN;
{Empty Queue. Tests if a queue is empty. }

procedure ENQUE (a,L: LIST);
{Enqueue. a is an object. L is a queue representing list. Appends a to the quque L. }

procedure FIRST (L: LIST): LIST;
{First. L is a non-null list. a is the first element of L. }

procedure NEWQUE (): LIST;
{New Queue. Returns a new empty queue. }

procedure INV (L: LIST): LIST;
{Inverse. L is a list. The inverse of L is returned. The list L is modified. }

procedure LENGTH (L: LIST): GAMMAINT;
{Length. L is a list. Returns length(L).}

procedure LIST1 (a: LIST): LIST;
{List, 1 element. a is an object. L is the list (a). }

procedure LISTVAR (VAR L: LIST);
{List variable. L is a list. The address of L is made accessible to the garbage collector. }

procedure RED (L: LIST): LIST;
{Reductum. L is a non-null list. Returns the reductum of L. }

procedure SFIRST (L, a: LIST);
{Set first. L is a non-null list. a is an object. The first element of L is changed to a. }

procedure SRED (L, LP: LIST);
{Set reductum. L is a non-null list. LP is a list. The reductum of L is changed to LP. }

procedure TIME (): GAMMAINT;
{Time. Returns the CLOCK minus the garbage collection time TAU. }

3.9 MAS mtc [Modula-2 to C]

procedure getstck (): ADDRESS;
{Get contents of stack register. }

procedure gettoc (): ADDRESS;
{Get contents of toc register. }

procedure NextParm (VAR s: ARRAY OF CHAR): BOOLEAN;
{Next Parameter. The next parameter from the GEM environement is placed in string s. If no parameter was found, FALSE is returned and s is undefined, else TRUE is returned. The separator for the parameters is the blank character. }

procedure DOS (s: ARRAY OF CHAR): INTEGER;
{Call DOS program. }

procedure EDIT (s: ARRAY OF CHAR): INTEGER;
{Edit file with name s. }

3.10 Portability

3.11 SAC Basic I/O System

procedure CWRIT2 (C1,C2: GAMMAINT);
{Character write, 2 characters. C1 and C2 are sequentially transmitted to the output stream using CWRITE.}

procedure CWRIT3 (C1,C2,C3: GAMMAINT);
{Character write, 3 characters. C1, C2 and C3 are sequentially transmitted to the output stream using CWRITE.}

procedure CWRIT4 (C1,C2,C3,C4: GAMMAINT);
{Character write, 4 characters. C1, C2, C3, and C4 are sequentially transmitted to the output stream using CWRITE.}

procedure CWRIT5 (C1,C2,C3,C4,C5: GAMMAINT);
{Character write, 5 characters. C1, C2, C3, C4 and C5 are sequentially transmitted to the output stream using CWRITE.}

procedure CWRIT6 (C1,C2,C3,C4,C5,C6: GAMMAINT);
{Character write, 6 characters. C1, C2, C3, C4, C5 and C6 are sequentially transmitted to the output stream using CWRITE.}

3.12 SAC List Processing

procedure ADV2 (L: LIST; VAR AL,BL,LP: LIST);
{Advance 2. L is a list of length two or more. a=FIRST(L), b=SECOND(L) and LP=RED(RED(L)).}

procedure ADV3 (L: LIST; VAR AL1,AL2,AL3,LP: LIST);
{Advance 3. L is a list of length 3 or more. a1, a2 and a3 are the first three elements of L. LP is the third reductum of L.}

procedure ADV4 (L: LIST; VAR AL1,AL2,AL3,AL4,LP: LIST);
{Advance 4. L is a list of length 4 or more. a1, a2, a3, and a4 are the first 4 elements of L. LP is the fourth reductum of L.}

procedure AREAD (): LIST;
{Atom read. An atom A is read from the input stream. Any preceding blanks are skipped.}

procedure AWRITE (A: LIST);
{Atom write. The atom A is written in the output stream.}

procedure CCONC (L1,L2: LIST): LIST;
{Constructive concatenation. L1 and L2 are lists. L is the concatenation of L1 and L2. The list L is constructed.}

procedure CINV (L: LIST): LIST;
{Constructive inverse. L is a list. M=INV(L). M is constructed using comp.}

procedure CLOUT (L: LIST);
{Character list out. The input is a character list L=(C(1),C(2),..., C(n)). The C(i) are sequentially transmitted to the output stream using CWRITE.}

procedure COMP2 (AL,BL,L: LIST): LIST;
{Composition 2. a and b are objects. L is a list. M=COMP(a,COMP(b,L)).}

procedure COMP3 (AL1,AL2,AL3,L: LIST): LIST;
{Composition 3. a1, a2 and a3 are objects. L is a list. M=COMP(a1,COMP(a2,COMP(a3,L))).}

procedure COMP4 (AL1,AL2,AL3,AL4,L: LIST): LIST;
{Composition 4. a1, a2, a3 and a4 are objects. L is a list. M=COMP(a1,COMP(a2,COMP(a3,COMP(a4,l)))).}

procedure CONC (L1,L2: LIST): LIST;
{Concatenation. L1 and L2 are lists. L=CONC(L1,L2). The list L1 is modified.}

procedure EQUAL (AL,BL: LIST): LIST;
{Equal. a and b are objects. t=1 if a and b are equal and otherwise t=0.}

procedure EXTENT (AL: LIST): LIST;
{Extent. a is an object. n=EXTENT(a).}

procedure FIRST2 (L: LIST; VAR AL,BL: LIST);
{First 2. L is a list of length 2 or more. a=FIRST(L) and b=SECOND(L).}

procedure FIRST3 (L: LIST; VAR AL1,AL2,AL3: LIST);
{First 3. L is a list of length 3 or more. a1=FIRST(L), a2=SECOND(L) and a3=THIRD(L).}

procedure FIRST4 (L: LIST; VAR AL1,AL2,AL3,AL4: LIST);
{First 4. L is a list of length 4 or more. a1=FIRST(L), a2=SECOND(L), a3=THIRD(L) and a4=FOURTH(L).}

procedure FOURTH (L: LIST): LIST;
{Fourth. L is a list of length 4 or more. a is the fourth element of L.}

procedure LAST (L: LIST): LIST;
{Last. L is a non-null list. LP is the location of the last cell of L.}

procedure LEINST (A,IL,AL: LIST): LIST;
{List element insertion. A is the list (a(1), ...,a(n)) of objects. i is a beta-integer, 0 le i le n. a is an object. If i=0, then L=(a,a(1), ...,a(n)). If i=n, then L=(a(1), ...,a(n),a). otherwise, L=(a(1), ...,a(i),a,a(i+1), ...,a(n)). A is modified.}

procedure LELT (A,IL: LIST): LIST;
{List element. A is a list. 1 le i le LENGTH(A). a is the i-th element of A.}

procedure LEROT (L,IL,JL: LIST): LIST;
{List element rotation. L is a list (a(1), ...,a(n)) of objects, n gt 0. i and j, 1 le i le j le n, are beta-integers. If i=j then M=L. Otherwise M=(a(1), ...,a(i-1),a(j),a(i), ...,a(j-1), a(j+1), ...,a(n)). L is modified.}

procedure LINS (AL,L: LIST);
{List insertion. L is a non-null list (a(1), ...,a(n)). a is an object. a is inserted in L after a(1) (suffixed to L if n=1), producing a modified list L=(a(1),a,a(2), ...,a(n)).}

procedure LINSRT (AL,A: LIST): LIST;
{List insertion. A is a list (a(1), ...,a(n)) of beta-integers, n ge 0, with a(1) lt a(2) lt ...lt a(n). If n=0 then B=(a). If a lt a(1) then B=(a,a(1), ...,a(n)). If a ge a(n) then B=(a(1), ...,a(n),a). Otherwise B=(a(1), ...,a(i),a,a(i+1), ...,a(n)) where a(i) le a lt a(i+1). The list A is modified to produce B.}

procedure LIST10 (AL1,AL2,AL3,AL4,AL5,AL6,AL7,AL8,AL9,AL10: LIST): LIST;
{List, 10 elements. a1, a2, a3, a4, a5, a6, a7, a8, a9 and a10 are objects. L is the list (a1,a2,a3,a4,a5,a6,a7,a8,a9,a10).}

procedure LIST2 (AL,BL: LIST): LIST;
{List, 2 elements. a and b are objects. L is the list (a,b).}

procedure LIST3 (AL1,AL2,AL3: LIST): LIST;
{List, 3 elements. a1, a2 and a3 are objects. L=(a1,a2,a3).}

procedure LIST4 (AL1,AL2,AL3,AL4: LIST): LIST;
{List, 4 elements. a1, a2, a3 and a4 are objects. L is the list (a1,a2,a3,a4).}

procedure LIST5 (AL1,AL2,AL3,AL4,AL5: LIST): LIST;
{List, 5 elements. a1,a2,a3,a4 and a5 are objects. L is the list (a1,a2,a3,a4,a5).}

procedure LREAD (): LIST;
{List read. The list L is read from the input stream. Any preceding blanks are skipped.}

procedure LSRCH (AL,A: LIST): LIST;
{List search. A is a list of beta-integers, (a(1), ...,a(n)), n ge 0. If there is a j such that a=a(j) then i is the least such j. Otherwise i=0.}

procedure LWRITE (L: LIST);
{List write. The input list L is written in the output stream.}

procedure MEMBER (AL,L: LIST): LIST;
{Membership test. a is an object, L a list. t=1 if a is a member of L and otherwise t=0.}

procedure ORDER (AL: LIST): LIST;
{Order. a is an object. n=ORDER(a).}

procedure OREAD (): LIST;
{Object read. The object B is read from the input stream. Any preceding blanks are skipped.}

procedure OWRITE (B: LIST);
{Object write. The input object B is written in the output stream.}

procedure PAIR (A,B: LIST): LIST;
{Pair. A=(a(1), ...,a(m)) and B=(b(1), ...,b(n)) are lists with m and n non-negative. C is the list (a(1),b(1), ...,a(r),b(r)) where r=MIN(m,n).}

procedure REDUCT (A,IL: LIST): LIST;
{Reductum. A is a list. i is a non-negative beta-integer not less than LENGTH(A). B=A, if i=0. Otherwise, B is the i-th reductum of A.}

procedure RED2 (L: LIST): LIST;
{Reductum 2. L is a list of length 2 or more. LP=RED(RED(L)).}

procedure RED3 (L: LIST): LIST;
{Reductum 3. L is a list of length 3 or more. M is the third reductum of L.}

procedure RED4 (L: LIST): LIST;
{Reductum 4. L is a list of length 4 or more. M is the fourth reductum of L.}

procedure SECOND (L: LIST): LIST;
{Second. L is a list of length 2 or more. a is the second element of L.}

procedure SLELT (A,IL,AL: LIST);
{Set list element. A is a list. 1 le i le LENGTH(A). The i-th element of A is changed to a.}

procedure SUFFIX (L,BL: LIST): LIST;
{Suffix. L is a list (a(1), ..., a(n)), n non-negative. b is an object. LP=(a(1), ..., a(n),b). L is modified.}

procedure THIRD (L: LIST): LIST;
{Third. L is a list of length 3 or more. a is the third element of L.}

3.13 System Informations

procedure SysInfoStart (VAR s:SYSINFO);
{system information start. The variable s needs no initialization. All data in s are lost. Informations over the usage of system resources are stored. Use this procedure to start the recording of statistical data over system resource usage. }

procedure SysInfoStop (VAR s:SYSINFO);
{system information stop. The variable s contains informations over system resources. The usage of this resources between the time of the initialization of s and now are stored in s. Use this procedure to stop the recording of statistical data over system resource usage. }

procedure SysInfoSum (a,b:SYSINFO; VAR s:SYSINFO);
{system information sum. The variable s needs no initialization. The sum of all data in a and b are stored in s. Use this procedure to join statistical data of two registration period. }

procedure SysInfoWrite (s: SYSINFO);
{system information write. The variable s contains informations over system resources. All informations in s are written out. Use this procedure to write out the statistical data of a registration }

3.14 clock

procedure ClocK (): LONGINT;
{ClocK returns milliseconds of the processes cpu-time. }

3.15 kpathsearch

procedure masReadOpen (file: ARRAY OF CHAR): IO_tFile;

{MAS Read Open file is the name of a file which is opened for reading using the kpse_path_search function of the kpathsearch library. The file is returned }

3.16 readline

procedure masReadL (VAR s: tString);
{MAS Read Line. Reads a line from StdInput and returns it in s using the GNU readline library. }

3.17 MAS Signal Handling

procedure signal (s: INTEGER; h: Action): Action;
{Set system signal handler. }

procedure raise (s: INTEGER): INTEGER;
{Raise signal s. }

procedure sigblock (mask: INTEGER):INTEGER;
{Block signals. }

procedure sigsetmask (mask: INTEGER): INTEGER;
{Set signal mask. }

procedure SigMask (s: INTEGER): INTEGER;
{Signal mask. }

3.18 Setjmp

procedure setjmp (VAR env: jmp_buf): INTEGER;
{Set jump environment. }

procedure longjmp (VAR env: jmp_buf; rc: INTEGER);
{Long jump to old environment. }

Chapter 4

LISP Algorithms

4.1 Aldes Parser

procedure Aparse (): LIST;
{Parse a set of ALDES-2 declarations and algorithms. }

4.2 MAS Lisp

procedure ECENV (ENV: LIST): LIST;
{Encode environment. The encoded environment E is returned. }

procedure DCENV (E: LIST): LIST;
{Decode environment. The encoded environment E is decoded. }

procedure SETV (V, A: LIST; VAR ENV: LIST);
{Set variable. V is a symbol and A is an S-expression. A is associated to V in the environment ENV. }

procedure EXTENDENV (A, X: LIST; VAR ENV: LIST): BOOLEAN;
{Extend environement. A is a list of symbols. X is a list of values. The environment ENV is extented by the bindings of the symbols in A to the values in X. If the binding is possible, then TRUE is returned else FALSE. }

procedure COPYTOENV (V, EP: LIST; VAR ENV: LIST);
{Copy to environement. V is a list of symbols. EP is an environment. The environment ENV is extented by the bindings of the symbols in V to the values in EP. }

procedure SPECIALFORM (S: LIST): BOOLEAN;
{Test if expression S is a special form. }

procedure LAMBDAP (S: LIST): BOOLEAN;
{Test if expression S is a lambda form. }

procedure SEXPRP (X: LIST): BOOLEAN;
{Test if X is a S-expression function. }

procedure DEFE (X: LIST; VAR ENV: LIST): LIST;
{Define expr function. X is a DE expression. A LAMBDA expression generated from X is associated to name(X) in the environment ENV. }

procedure DEFF (X: LIST; VAR ENV: LIST): LIST;
{Define fexpr function. X is a DF expression. A FLAMBDA expression generated from X is associated to name(X) in the environment ENV.}

procedure DEFM (X: LIST; VAR ENV: LIST): LIST;
{Define macro function. X is a DM expression. A MLAMBDA expression generated from X is associated to name(X) in the environment ENV.}

procedure DEFMAP (X: LIST; VAR ENV: LIST): LIST;
{Define generic map function. X is a MAP expression. A GLAMBDA expression generated from X is associated to name(X) in the environment ENV. }

procedure DEFPROC (X: LIST; VAR ENV: LIST): LIST;
{Define generic proc function. X is a DE expression. A GLAMBDA expression generated from X is associated to name(X) in the environment ENV.}

procedure DEFRULE (X: LIST; VAR ENV: LIST): LIST;
{Define generic rule function. X is a RULE expression. A GLAMBDA expression generated from X is associated to name(X) in the environment ENV.}

procedure DSPEC (X: LIST; VAR ENV: LIST): LIST;
{Define specification. X is a SPEC expression. An UNIT expression generated from X is associated to name(X) in the environment ENV.}

procedure DMIA (X: LIST; VAR ENV: LIST): LIST;
{Define model, implementation or axioms. X is a MODEL, IMPLEMENTATION or AXIOMS expression. An UNIT expression associated to name(X) is modified by a generated expression of X. }

procedure TYPEOF (X: LIST): LIST;
{Type of S-expression. X is an S-expression. A list of types, values and descriptors of X is returend. }

procedure TAG (V,T: LIST): LIST;
{Tag object. V is an S-expression, T is a type expression. A tagged TYPEINFO S-expression is returned. }

procedure VALOFTAG (L: LIST): LIST;
{Value of tagged object. L is a tagged S-expression. The value component of L is returned. }

procedure TYPOFTAG (L: LIST): LIST;
{Type of tagged object. L is a tagged S-expression. The type component of L is returned.}

procedure DECOFTAG (L: LIST): LIST;
{Descriptor of tagged object. L is a tagged S-expression. The descriptor component of L is returned. }

procedure GENPL (P,V,T,D: LIST): LIST;
{Generate parameter list. P is a list of patterns. V is a list of values. T is a list of types and D is a list of descriptors. A parameter list is returned. }

procedure GENTE (Z,N,D: LIST): LIST;
{Generate typed expression. Z is an S-expression, N is a function name, D is a descriptor. A typed S-expression for evaluation is returned. }

4.3 MAS Lisp Utility

procedure CallCompiled (F, PI: LIST; VAR PO: LIST; VAR fu: BOOLEAN);
{Call compiled function or procedure. F is a function or procedure symbol. PI is the list of input parameters. fu is TRUE if F is a function and FALSE if F is a procedure. PO is a list of output parameters if F is a procedure and PO is the output parameter if F is a function. }

procedure Compiledp0 (F: PROCP0; VAR S: ARRAY OF CHAR);
{Compiled function declaration p0. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledp1 (F: PROCP1; VAR S: ARRAY OF CHAR);
{Compiled function declaration p1. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledp2 (F: PROCP2; VAR S: ARRAY OF CHAR);
{Compiled function declaration p2. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledp3 (F: PROCP3; VAR S: ARRAY OF CHAR);
{Compiled function declaration p3. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledp4 (F: PROCP4; VAR S: ARRAY OF CHAR);
{Compiled function declaration p4. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledp5 (F: PROCP5; VAR S: ARRAY OF CHAR);
{Compiled function declaration p5. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledf0 (F: PROCF0; VAR S: ARRAY OF CHAR);
{Compiled function declaration f0. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledf1 (F: PROCF1; VAR S: ARRAY OF CHAR);
{Compiled function declaration f1. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledf2 (F: PROCF2; VAR S: ARRAY OF CHAR);
{Compiled function declaration f2. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledf3 (F: PROCF3; VAR S: ARRAY OF CHAR);
{Compiled function declaration f3. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledf4 (F: PROCF4; VAR S: ARRAY OF CHAR);
{Compiled function declaration f4. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledf5 (F: PROCF5; VAR S: ARRAY OF CHAR);
{Compiled function declaration f5. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledp1v2 (F: PROCP1V2; VAR S: ARRAY OF CHAR);
{Compiled function declaration p1v2. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledp1v3 (F: PROCP1V3; VAR S: ARRAY OF CHAR);
{Compiled function declaration p1v3. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledp2v2 (F: PROCP2V2; VAR S: ARRAY OF CHAR);
{Compiled function declaration p2v2. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledp2v3 (F: PROCP2V3; VAR S: ARRAY OF CHAR);
{Compiled function declaration p2v3. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledp3v2 (F: PROCP3V2; VAR S: ARRAY OF CHAR);
{Compiled function declaration p3v2. F is a Modula-2 procedure, S is the print name of F. }

procedure Compiledp3v3 (F: PROCP3V3; VAR S: ARRAY OF CHAR);
{Compiled function declaration p3v3. F is a Modula-2 procedure, S is the print name of F. }

procedure CompSummary ;
{Compiled function and procedure summary. Write out all compiled functions with their signature from symbol table SYMTB. }

procedure Declare (VAR X: LIST; VAR S: ARRAY OF CHAR);
{Declare. X is declared as symbol with print name S. }

procedure PROCP (X: LIST): BOOLEAN;
{Procedure Pointer. Test if the symbol X is a compiled function. }

procedure Signature (F: LIST; VAR PI, PO: LIST; VAR def: BOOLEAN);
{Signature of a compiled function or procedure. F is a function or procedure symbol. PI is the number of input parameters. def is TRUE if F is defined as compiled function or procedure else

def is FALSE. PO is the number of output parameters if F is a procedure, PO = -1 if F is a function. }

4.4 MAS Parser

procedure Parse (): LIST;
{Parse program and generate code. }

procedure SwitchParse (g: BOOLEAN);
{Switch parsing between generic / non-generic parse. If g = TRUE then the parser generates code for generic names, if g = FALSE then the parser generates code for the builtin LISP arithmetic functions. }

4.5 MAS Representation

procedure NewRep (): LIST;
{New representation. A new representation is returned. }

procedure SetRep (n,e,r: LIST);
{Set representation. r is a representation. n is a unique label, e is an expression. }

procedure GetRep (n,r: LIST): LIST;
{Get representation. r is a representation. n is a unique label, An expression e stored under n is returned. }

procedure CopyRep (r: LIST): LIST;
{Copy representation. r is a representation. A copy of r is returned. Step counters are reset. }

procedure StepRep (r: LIST): LIST;
{Step through representation. r is a representation. The next expression e and its label n are returned as pair (n,e). If all pairs (n,e) of representation r have been processed, then NIL is returned. }

procedure ForEachinRep (r, f, E: LIST): LIST;
{For each pair (n,e) in r apply function f. r is a representation. (n, e) is a pair in r, where e is an expression and n is its label. f is applied to (n, e) such that p = f(n,e) with p = (n', e') or p = NIL. If p ¡¿ NIL, then the pair (n', e') is added to the returned representation r' = n' e' : f(n,e) ¡¿ NIL, n e in r . E is a LISP environment, if E = NIL, then the top level environment is used. }

procedure FullRep (r: LIST): LIST;
{Full representation. Test for non-empty representation. r is a representation. }

procedure ForEachinList (r, f, E: LIST): LIST;
{For each element e in r apply function f. r is a list. e is an element in r. f is applied to e such that e' = f(e). E is a LISP environment, if E = NIL, then the top level environment is used. }

4.6 MAS Specification

procedure EVALUATE (X: LIST; VAR ENV: LIST): LIST;
{Lisp evaluator. X is an S-expression, ENV is an environment. }

4.7 MAS Symbol

procedure ATOM (X: LIST): BOOLEAN;
{Atom. Test if X is an atom. }

procedure ELEMP (X: LIST): BOOLEAN;
{Elementary Pointer. Test if X is an elementary SAC-2 structure. }

procedure MEMQ (AL,L: LIST): BOOLEAN;
{Membership test equal pointers. a is an object, L a list. t=1 if the pointer or atom a occurs in L and otherwise t=0.}

procedure OCCURQ (AL,L: LIST): BOOLEAN;
{Occurs test equal pointers. a and L are objects. t=TRUE if the pointer or atom a occurs in L and otherwise t=FALSE. }

procedure UREAD (): LIST;
{Universal read. The next atom, symbol, string or list over atoms, strings and symbols is read and stored under L. Blanks may occur anywhere. Elements of a list may or may not be separated by a comma.}

procedure UWRITE (L: LIST);
{Universal write. L is an atom, symbol or a list over atoms and symbols. L is written in the output stream, followed by blines. }

procedure UWRIT1 (L: LIST);
{Universal write, 1. subalgorithm. L is an atom, a symbol or a list over atoms or symbols. L is written in the output stream. }

procedure UNIFY (A,B: LIST; VAR S: LIST): BOOLEAN;
{Unification. A and B are objects. If there exists a most general unificator of A and B, then S is the list of substitutions. In this case TRUE is returned. If no unificator exists, then FALSE is returned and S is undefined. }

procedure GENARRAY (A: LIST): LIST;
{Generate array reference symbol. S is a generated symbol. }

procedure GENINDEX (A: LIST): LIST;
{Generate index set. I is an index set. }

procedure ARRAYDEC (A: LIST): LIST;
{Generate array name declarations. A is an array reference. }

4.8 MAS/SAC Symbol System 2.

procedure ACOMP (A,B: LIST): LIST;
{Alphabetic comparison. A and B are symbols. t=+1,0,-1 according to whether A preceds, is equal, or follows B alphabetically.}

procedure ACOMP1 (A,B: LIST): LIST;
{Alphabetic comparison, 1. subalgorithm. A and B are packed strings. s=-1,0,1 according to whether a preceds, is equal, or succeeds B alphabetically.}

procedure ASSOC (AL,L: LIST): LIST;
{Associate. L=(a1 b1, a2 b2, ...,a sub n b sub n), n ge 0, a is an object. If there is an i such that a=a sub i then P=(b sub i, ...,a sub n b sub n), otherwise P=().}

procedure ASSOCQ (AL,L: LIST): LIST;
{Associate equal. L=(a1 b1, a2 b2, ...,a sub n b sub n), n ge 0, a is an object. If there is an i such that a is equal to a sub i then P=(b sub i, ...,a sub n b sub n), otherwise P=().}

procedure ATTRIB (L: LIST): LIST;
{Attribute. L is a symbol. Returns the attributes of L.}

procedure EXPLOD (S: LIST): LIST;
{Explode symbol. S is a symbol, L its character list.}

procedure ENTER (L: LIST): LIST;
{Enter into symbol table. L is a character list, S the pointer to the corresponding symbol. If the symbol is not yet in the symbol table SYMTB, then a new node is created.}

procedure GENSYM (): LIST;
{Generate symbol. S is a newly generated symbol. NAM is advanced.}

procedure GET (S,AL: LIST): LIST;
{Get property. The property list of the symbol S is searched under indicator a. A is the property under a, if any, otherwise A is set to beta.}

procedure NAME (L: LIST): LIST;
{Name. L is a symbol. Returns the name of L.}

procedure PACK (L: LIST): LIST;
{Pack character list. L is a non-empty character list. B is the packed list.}

procedure PUT (S,AL,A: LIST);
{Put. The property A is stored on the property list of the symbol S under the indicator a.}

procedure REMPRP (S,AL: LIST);
{Remove property. Under indicator a on the property list of symbol S the property is removed.}

procedure SMEMB (S,L: LIST): LIST;
{Symbol membership. S is a symbol, L a list containing possibly also symbols. b=1 if S or a copy of S occurs in L, b=0 otherwise.}

procedure SREAD (): LIST;
{Symbol read. The next symbol is read from input. S is the symbol in the symbol table SYMTB.}

procedure SREAD1 (): LIST;
{Symbol read, 1. The first non-alphanumeric character of the input stream terminates the symbol. L is the character list of the symbol, which is not entered in the symbol table.}

procedure STCNT (T: LIST; VAR S,P: LIST);
{Symbol table tree count. T is a symbol tree, S is the number of symbols in T, P the number of properties of all symbols of the tree. Since every symbol has a name property, P ge S.}

procedure STINS (B: LIST): LIST;
{Symbol tree insertion. B is a packed list of characters. S is a pointer to the corresponding symbol in the symbol table. If it is not yet in, a new node is created.}

procedure STLST (T: LIST): LIST;
{Symbol tree list. T is a symbol tree, L is the list of its symbols in alphabetic order.}

procedure STLSTI (T: LIST): LIST;
{Symbol tree list, in-order. T is a binary tree of symbols, L is a list of its symbols, with the root symbol appearing first.}

procedure STSRCH (T,AP: LIST): LIST;
{Symbol tree search. T is a binary tree of symbols, AP is a packed list of characters. If the symbol with the name AP occurs already in the symbol table T then S=() and otherwise S points to the entry.}

procedure STWRT (T: LIST);
{Symbol tree write. T is a binary tree of symbols. The symbols followed by their properties are printed in alphabetic order.}

procedure SYMBOL (AP: LIST): BOOLEAN;
{Symbol. AP is an object. Returns true if it is a symbol and false else.}

procedure SymSummary ();
{Summary of symbol system. The number of symbols in SYMTB and the number of their properties is written.}

procedure SYWRIT (S: LIST);
{Symbol write. The symbol S is written in the output stream.}

procedure SUBLIS (L,A: LIST): LIST;
{Substitution with list. L=(x1 e1, ...,x sub n e sub n), a and e sub i are objects. The x sub i are beta-digits or pointers to uniquely stored lists like symbols. B is A with the x sub i substituted by the e sub i.}

procedure UREAD (): LIST;
{Universal read. The next atom, symbol or list over atoms and symbols is read and stored under L. Blanks may occur anywhere, elements of a list may or may not be separated by a comma.}

procedure UWRITE (L: LIST);
{Universal write. L is an atom, symbol or a list over atoms and symbols. L is written in the output stream, followed by BLINES(0). }

procedure UWRIT1 (L: LIST);
{Universal write, 1. subalgorithm. L is an atom, a symbol or a list over atoms or symbols. L is written in the output stream followed by a blank character, but not by BLINES. }

4.9 Modula Global Variable Implementation Module.

procedure MVDeclareL (VAR var: LIST; name,comment: ARRAY OF CHAR;
access: BOOLEAN); {modula variable declare list. The global variable var is made accessible for the MAS interpreter under the name name. Comment is a string which explains the variable. access is a flag that determines, whether var is protected from overwriting or not. }

procedure MVDeclareB (VAR var: BOOLEAN; name,comment: ARRAY OF CHAR;
access:BOOLEAN); {modula variable declare boolean. The global variable var is made accessible for the MAS interpreter under the name name. Comment is a string which explains the variable. access is a flag that determines, whether var is protected from overwriting or not. }

procedure MVSET (sym,value:LIST);
{modula variable set. sym is a symbol, value is a list. The value value is assigned to the modula variable with the interpreter-name sym. }

procedure MVGET (sym:LIST): LIST;
{modula variable get. The value of the modula variable with the name sym is returned. }

procedure MVFLAG (sym:LIST): LIST;
{modula variable get. The boolean value of the modula variable with the name sym is returned. TRUE is equivalent to 1 and FALSE is equivalent to 0. }

procedure MVON (sym:LIST);
{modula variable on. The value 1 is assigned to the module variable with the interpreter name sym. }

procedure MVOFF (sym:LIST);
{modula variable off. The value 0 is assigned to the module variable with the interpreter name sym. }

procedure MVHLP (sym:LIST);
{modula variable help. All known informations over the modula variable with the interpreter name sym is printed to the output stream. }

4.10 SAC Symbol System

procedure ACOMP (A,B: LIST): LIST;
{Alphabetic comparison. A and B are symbols. t=+1,0,-1 according to whether A preceds, is equal, or follows B alphabetically.}

procedure ACOMP1 (A,B: LIST): LIST;
{Alphabetic comparison, 1. subalgorithm. A and B are packed strings. s=-1,0,1 according to whether a preceds, is equal, or succeeds B alphabetically.}

procedure ASSOC (AL,L: LIST): LIST;
{Associate. L=(a1 b1, a2 b2, ...,a sub n b sub n), n ge 0, a is an object. If there is an i such that a=a sub i then P=(b sub i, ...,a sub n b sub n), otherwise P=().}

procedure ASSOCQ (AL,L: LIST): LIST;
{Associate equal. L=(a1 b1, a2 b2, ...,a sub n b sub n), n ge 0, a is an object. If there is an i such that a is equal to a sub i then P=(b sub i, ...,a sub n b sub n), otherwise P=().}

procedure ATTRIB (L: LIST): LIST;
{Attribute. L is a symbol. Returns the attributes of L.}

procedure EXPLOD (S: LIST): LIST;
{Explode symbol. S is a symbol, L its character list.}

procedure ENTER (L: LIST): LIST;
{Enter into symbol table. L is a character list, S the pointer to the corresponding symbol. If the symbol is not yet in the symbol table SYMTB, then a new node is created.}

procedure GENSYM (): LIST;
{Generate symbol. S is a newly generated symbol. NAM is advanced.}

procedure GET (S,AL: LIST): LIST;
{Get property. The property list of the symbol S is searched under indicator a. A is the property under a, if any, otherwise A is set to beta.}

procedure NAME (L: LIST): LIST;
{Name. L is a symbol. Returns the name of L.}

procedure PACK (L: LIST): LIST;
{Pack character list. L is a non-empty character list. B is the packed list.}

procedure PUT (S,AL,A: LIST);
{Put. The property A is stored on the property list of the symbol S under the indicator a.}

procedure REMPRP (S,AL: LIST);
{Remove property. Under indicator a on the property list of symbol S the property is removed.}

procedure SMEMB (S,L: LIST): LIST;
{Symbol membership. S is a symbol, L a list containing possibly also symbols. b=1 if S or a copy of S occurs in L, b=0 otherwise.}

procedure SREAD (): LIST;
{Symbol read. The next symbol is read from input. S is the symbol in the symbol table SYMTB.}

procedure SREAD1 (): LIST;
{Symbol read, 1. The first non-alphanumeric character of the input stream terminates the symbol. L is the character list of the symbol, which is not entered in the symbol table.}

procedure STCNT (T: LIST; VAR S,P: LIST);
{Symbol table tree count. T is a symbol tree, S is the number of symbols in T, P the number of properties of all symbols of the tree. Since every symbol has a name property, P ge S.}

procedure STINS (B: LIST): LIST;
{Symbol tree insertion. B is a packed list of characters. S is a pointer to the corresponding symbol in the symbol table. If it is not yet in, a new node is created.}

procedure STLST (T: LIST): LIST;
{Symbol tree list. T is a symbol tree, L is the list of its symbols in alphabetic order.}

procedure STLSTI (T: LIST): LIST;
{Symbol tree list, in-order. T is a binary tree of symbols, L is a list of its symbols, with the root symbol appearing first.}

procedure STSRCH (T,AP: LIST): LIST;
{Symbol tree search. T is a binary tree of symbols, AP is a packed list of characters. If the symbol with the name AP occurs already in the symbol table T then S=() and otherwise S points to the entry.}

procedure STWRT (T: LIST);
{Symbol tree write. T is a binary tree of symbols. The symbols followed by their properties are printed in alphabetic order.}

procedure SYMBOL (AP: LIST): BOOLEAN;
{Symbol. AP is an object. Returns true if it is a symbol and false else.}

procedure SymSummary ();
{Summary of symbol system. The number of symbols in SYMTB and the number of their properties is written.}

procedure SYWRIT (S: LIST);
{Symbol write. The symbol S is written in the output stream.}

procedure SUBLIS (L,A: LIST): LIST;
{Substitution with list. L=(x1 e1, ...,x sub n e sub n), a and e sub i are objects. The x sub i are beta-digits or pointers to uniquely stored lists like symbols. B is A with the x sub i substituted by the e sub i.}

procedure UREAD (): LIST;
{Universal read. The next atom, symbol or list over atoms and symbols is read and stored under L. Blanks may occur anywhere, elements of a list may or may not be separated by a comma.}

procedure UWRITE (L: LIST);
{Universal write. L is an atom, symbol or a list over atoms and symbols. L is written in the output stream, followed by BLINES(0). }

procedure UWRIT1 (L: LIST);
{Universal write, 1. subalgorithm. L is an atom, a symbol or a list over atoms or symbols. L is written in the output stream followed by a blank character, but not by BLINES. }

4.11 SAC Symbol 2

procedure STBAL (L,n: LIST): LIST;
{Symbol tree balance. L is an alphabetical list of n symbol-tree nodes (n gt 0), out of which a balanced binary tree S is constructed. }

procedure STBALS (VAR A: ARRAY OF LIST; l, r: INTEGER): INTEGER;
{Symbol tree balance subroutine. The array A contains symbol-tree nodes in alphabetical order. The binary tree of the symbols in A[l..r] is constructed and A[m] is its root. }

procedure STNLST (T: LIST; VAR L,n: LIST);
{Symbol tree nodes list. T is a non-empty symbol tree. L is the list of its nodes in alphabetical order of the corresponding symbols and n the number of nodes. This algorithm is normally used for creating the data as required for STBAL. }

procedure SSYTBAL ;
{System symbol tree balance. SYMTB is balanced. }

Chapter 5

Main Algorithms

5.1 MAS Load

procedure InitExternals ;
{Initialize external compiled procedures. }

5.2 MAS Load A.

procedure InitExternalsA ;
{Initialize external compiled arithmetic procedures. }

5.3 MAS Load B.

procedure InitExternalsB ;
{Initialize external compiled polynomial procedures. }

5.4 MAS Load C.

procedure InitExternalsC ;
{Initialize external compiled non-commutative polynomial procedures. }

5.5 MAS Load D.

procedure InitExternalsD ;
{Initialize external compiled ideal decomposition and root procedures. }

5.6 MAS Load E.

procedure InitExternalsE ;
{Initialize external compiled arbitrary domain procedures. }

5.7 MAS Load Symmetric Functions

procedure InitExternalsG ;
{Tell Modula and LISP about external compiled procedures. }

5.8 MAS Load J.

procedure InitExternalsJ ;
{Initialize external compiled arithmetic procedures. }

5.9 MAS Load L.

procedure InitExternalsL ();
{Initialize external compiled linear algebra procedures. }

5.10 MAS Load M.

procedure InitExternalsM ;
{Initialize external compiled real root procedures. }

5.11 MAS Load Q.

procedure InitExternalsQ ;
{Initialize external compiled arithmetic Q procedures. }

5.12 MAS Load Syzygy

procedure InitExternalsS ;
{Tell Modula and LISP about external compiled procedures. }

5.13 MAS Utility

procedure InitExternalsU ();
{Initialize external compiled utility procedures. }

procedure DoParse (): LIST;
{Do parse. Call specific Parser. }

procedure DoWrite (Y: LIST);
{Do Write. Write according to Parser. }

procedure MWRITE (Y: LIST);
{Output in modula like syntax. }

procedure MWRIT1 (Y: LIST; top: BOOLEAN);
{Output in modula like syntax. }

5.14 MAS Symbol to DIP

procedure DIPVDEF (V: LIST): LIST;
{DIP define distributive polynomial variable list. V is a variable list. The new variable list is returned. }

procedure DIPTODEF (T: LIST): LIST;
{DIP define distributive polynomial term order. V is a term order indicator. The old term order indicator is returned. }

procedure SYM2DIP (T: LIST): LIST;
{Symbol term to distributive polynomial. }

procedure DIP2SYM (D: LIST): LIST;
{Distributive polynomial to symbol term. }

procedure TVARS (T: LIST): LIST;
{Term variables. T is a term. The list of variables occuring in T is returned. }

procedure DIRPFT (T, V: LIST): LIST;
{Distributive rational polynomial from term. T is a term, V is a variable list. A distributive rational polynomial A in r variables, where r=length(V), r ge 0, is formed from term T. }

procedure TFDIRP (A, V: LIST): LIST;
{Term from distributive rational polynomial. A is a distributive rational polynomial in r variables, where r=length(V), r ge 0, V is a symbol list. A term T is formed from A. }

procedure InitExternalsI ;
{Initialize external compiled interface procedures. }

5.15 MAS Logic Configuration Implementation Module.

procedure InitExternalsML ;
{Initialze external compiled logic procedures. }

5.16 MAS Logic Demonstration Implementation Module.

procedure InitExternalsMLDEMO ();
{Initialize externals maslog demonstration procedures. }

5.17 Masload Polynomial Equation Simplify

procedure InitExternalsPQSMPL ();
{initialize external compiled PQS-procedures. }

Chapter 6

Arithmetic Algorithms

6.1 MAS Arbitrary Precision Floating Point

procedure APCOMP (ML,EL: LIST): LIST;
{Arbitrary precision floating point composition. e is the exponent, m is the mantissa of the arbitrary precision floating point number A.}

procedure APMANT (A: LIST): LIST;
{Arbitrary precision floating point mantissa. m is the mantissa of the arbitrary precision floating point number A.}

procedure APEXPT (A: LIST): LIST;
{Arbitrary precision floating point exponent. e is the exponent of the arbitrary precision floating point number A.}

procedure ILOG10 (N: LIST): LIST;
{Integer logarithm base 10. N is an integer, l is a beta integer. l=LOG10(ABS(N)).}

procedure APSPRE (N: LIST);
{Arbitrary precision floating point set precision. N is the desired precision of the floating point numbers.}

procedure APFINT (N: LIST): LIST;
{Arbitrary precision floating point from integer. The integer N is converted to the arbitrary precision floating point number A.}

procedure APSHFT (B,EL: LIST): LIST;
{Arbitrary precision floating point shift. The arbitrary precision floating point number B is multiplied by 2**e. A is an arbitrary precision floating point number.}

procedure APSIGN (A: LIST): LIST;
{Arbitrary precision floating point sign. A is an arbitrary precision floating point number, s is the sign of A.}

procedure APWRIT (A: LIST);
{Arbitrary precision floating point write. The arbitrary precision floating point number A is written to the output stream.}

procedure APNEG (A: LIST): LIST;
{Arbitrary precision floating point negative. The arbitrary precision floating point number A is negated. B= -A.}

procedure APABS (A: LIST): LIST;
{Arbitrary precision floating point absolute value. A is a arbitrary precision floating point number. B is the absolute value of A.}

procedure APCMPR (A,B: LIST): LIST;
{Arbitrary precision floating point compare. A and B are arbitrary precision floating point numbers. s is the sign of the difference of A and B. s=SIGN(A-B).}

procedure APNELD (A,B: LIST): LIST;
{Arbitrary precision floating point number of equal leading digits. A and B are arbitrary precision floating point numbers. l is the number of equal leading digits of A and B.}

procedure APPROD (A,B: LIST): LIST;
{Arbitrary precision floating point product. A, B and C are arbitrary precision floating point numbers. C is the product of A and B. C=A*B.}

procedure APQ (A,B: LIST): LIST;
{Arbitrary precision floating point quotient. A, B and C are arbitrary precision floating point numbers. C is the quotient of A and B. C=A/B.}

procedure APSUM (A,B: LIST): LIST;
{Arbitrary precision floating point sum. A, B and C are arbitrary precision floating point numbers. C is the sum of A and B. C=A+B.}

procedure APDIFF (A,B: LIST): LIST;
{Arbitrary precision floating point difference. A, B and C are arbitrary precision floating point numbers. C is the difference of A and B. C=A-B.}

procedure APLG10 (A: LIST): LIST;
{Arbitrary precision floating point logarithm base 10. A is an arbitrary precision floating point number, l is a beta integer, l=LOG10(ABS(A)). }

procedure APEXP (A,NL: LIST): LIST;
{Arbitrary precision floating point exponentiation. A and B are arbitrary precision floating point numbers. n is a beta-integer. B=A**n.}

procedure APFRN (A: LIST): LIST;
{Arbitrary precision floating point from rational number. B is an arbitrary precision floating point number. A is a rational number.}

procedure RNFAP (A: LIST): LIST;
{Rational number from arbitrary precision floating point. A is an arbitrary precision floating point number. B is a rational number.}

procedure RNDRD (): LIST;
{Rational number decimal read. The rational number R is read from the input stream. Any preceding blanks are skipped.}

procedure APROOT (A,NL: LIST): LIST;
{Arbitrary precision floating point n-th root. A and B are arbitrary precision floating point numbers. B is the n-th root of A.}

procedure APPI (): LIST;
{Arbitrary precision floating point pi. pi is an arbitrary precision floating point number. }

6.2 MAS Complex Number

procedure CABS (R: LIST): LIST;
{Complex number absolute value. R is a complex number. S is the absolute value of R, a rational number. }

procedure CCON (R: LIST): LIST;
{Complex number conjugate. R is a complex number. S is the complex conjugate of R. }

procedure CCOMP (R,S: LIST): LIST;
{Complex number comparison. R and S are complex numbers. t=0 if R=S, t=1 else. }

procedure CDIF (R,S: LIST): LIST;
{Complex number difference. R and S are complex numbers. T=R-S. }

procedure CDREAD (): LIST;
{Complex number decimal read. The complex number R is read from the input stream. Any preceding blanks are skipped. }

procedure CDWRITE (R,NL: LIST);
{Complex number decimal write. R is a complex number. n is a non-negative integer. R is approximated by a decimal fraction D with n decimal digits following the decimal point and D is written in the output stream. The inaccuracy of the approximation is at most (1/2)*10**-n. }

procedure CEXP (A,NL: LIST): LIST;
{Complex number exponentiation. A is a complex number, n is a non-negative beta-integer. B=A**n. }

procedure CIM (R: LIST): LIST;
{Complex number imaginary part. R is a complex number. b is the imaginary part of R, a rational number. }

procedure CINT (A: LIST): LIST;
{Complex number from integer. A is an integer. R is the complex number with real part A/1 and imaginary part 0. }

procedure CRE (R: LIST): LIST;
{Complex number real part. R is a complex number. b is the real part of R, a rational number. }

procedure CRN (A: LIST): LIST;
{Complex number from rational number. A is a rational number. R is the complex number with real part A and imaginary part 0. }

procedure CRNP (A, B: LIST): LIST;
{Complex number from pair of rational numbers. A and B are rational numbers. R is the complex number with real part A and imaginary part B. }

procedure CNINV (R: LIST): LIST;
{Complex number inverse. R is a non-zero complex number. S R=1. }

procedure CNEG (R: LIST): LIST;
{Complex number negative. R is a complex number. S=-R. }

procedure CONE (R: LIST): LIST;
{Complex number one. R is a complex number. s=1 if R=1, s=0 else. }

procedure CPROD (R,S: LIST): LIST;
{Complex number product. R and S are complex numbers. T=R*S. }

procedure CQ (R,S: LIST): LIST;
{Complex number quotient. R and S are complex numbers, S non-zero. T=R/S. }

procedure CRAND (NL: LIST): LIST;
{Complex number, random. n is a positive beta-integer. Random rational numbers A and B are generated using RNRAND(n). Then R is the complex number with real part A and imaginary part B. }

procedure CNREAD (): LIST;
{Complex number read. The complex number R is read from the input stream. Any preceding blanks are skipped. }

procedure CSUM (R,S: LIST): LIST;
{Complex number sum. R and S are complex numbers. T=R+S. }

procedure CNWRITE (R: LIST);
{Complex number write. R is a complex number. R is converted to decimal and written in the output stream. }

6.3 MAS Combinatorial System

procedure INVPERM (perm: LIST):LIST;
{inverse permutation. perm is a permutation. The inverse permutation is returned, i.e. LPERM(LPERM(x,p),INVPERM(p))=x. }

procedure PVFLISTS (list1,list2:LIST):LIST;
{permutation vector from lists. list1 and list2 are lists of the same length. A permutations vector P is returned, so that LPERM(list1,P)=list2 }

6.4 MAS Floating Point

procedure FFGI (N: GAMMAINT): MFLOAT;
{Floating point from gamma integer. The gamma integer N is converted to the floating point number A. }

procedure IFF (F: MFLOAT): LIST;
{Integer from floating point. The floating point number F is converted to the integer A. }

procedure FEXP (F: MFLOAT; N: GAMMAINT): MFLOAT;
{Floating point exponentiation. The floating point number F raised to the n-th power. }

procedure FLOG10 (F: MFLOAT): MFLOAT;
{Floating point logarithm base 10. The logarithm of the floating point number F with base 10 is returned. }

procedure FFINT (N: LIST): MFLOAT;
{Floating point from integer. The integer N is converted to the floating point number f. }

procedure FFRN (A: LIST): MFLOAT;
{Floating point from rational number. The rational number A is converted to the floating point number f. }

procedure RNFF (F: MFLOAT): LIST;
{Rational number from floating point. The floating point number F is converted to the rational number R. }

procedure SIN (A: LIST): LIST;
{Sinus. A is a rational number, the sinus of A is returned. }

procedure COS (A: LIST): LIST;
{Cosinus. A is a rational number, the cosinus of A is returned. }

procedure TAN (A: LIST): LIST;
{Tangens. A is a rational number, the tangens of A is returned. }

procedure ARCTAN (A: LIST): LIST;
{Arcus tangens. A is a rational number, the arctangens of A is returned. }

procedure EXPF (A: LIST): LIST;
{Exponential. A is a rational number, the exponential of A is returned. }

procedure LN (A: LIST): LIST;
{Ln. A is a rational number, the natural logarithm of A is returned. }

procedure LOG (A: LIST): LIST;
{Log. A is a rational number, the logarithm base 10 of A is returned. }

procedure SQRT (A: LIST): LIST;
{Sqrt. A is a rational number, the square root of A is returned. }

6.5 MAS Integer

procedure IPROD (A,B: LIST): LIST;
{Integer product. A and B are integers. C=A*B. For long integers Karatsubas method is used. }

6.6 MAS Octonion Number

procedure OABS (R: LIST): LIST;
{Octonion number absolute value. R is a octonion number. S is the absolute value of R, a rational number. }

procedure OCON (R: LIST): LIST;
{Octonion number conjugate. R is a octonion number. S is the octonion conjugate of R. }

procedure OCOMP (R,S: LIST): LIST;
{Octonion number comparison. R and S are octonion numbers. t=0 if R=S, t=1 else. }

procedure ODIF (R,S: LIST): LIST;
{Octonion number difference. R and S are octonion numbers. T=R-S. }

procedure ODREAD (): LIST;
{Octonion number decimal read. The octonion number R is read from the input stream. Any preceding blanks are skipped. }

procedure ODWRITE (R,NL: LIST);
{Octonion number decimal write. R is a octonion number. n is a non-negative integer. R is approximated by a decimal fraction D with n decimal digits following the decimal point and D is written in the output stream. The inaccuracy of the approximation is at most (1/2)*10**-n. }

procedure OEXP (A,NL: LIST): LIST;
{Octonion number exponentiation. A is a octonion number, n is a non-negative beta-integer. B=A**n. }

procedure OIM (R: LIST): LIST;
{Octonion number imaginary part. R is a octonion number. b is the imaginary part of R, a rational number. }

procedure OINT (A: LIST): LIST;
{Octonion number from integer. A is an integer. R is the octonion number with real part A/1 and imaginary part 0. }

procedure ORE (R: LIST): LIST;
{Octonion number real part. R is a octonion number. b is the real part of R, a rational number. }

procedure ORN (A: LIST): LIST;
{Octonion number from rational number. A is a rational number. R is the octonion number with real part A and imaginary part 0. }

procedure ORNP (A, B: LIST): LIST;
{Octonion number from pair of rational numbers. A and B are rational numbers. R is the octonion number with real part A and imaginary part B. }

procedure ONINV (R: LIST): LIST;
{Octonion number inverse. R is a non-zero octonion number. S R=1. }

procedure ONEG (R: LIST): LIST;
{Octonion number negative. R is a octonion number. S=-R. }

procedure OONE (R: LIST): LIST;
{Octonion number one. R is a octonion number. s=1 if R=1, s=0 else. }

procedure OPROD (R,S: LIST): LIST;
{Octonion number product. R and S are octonion numbers. T=R*S. }

procedure OQ (R,S: LIST): LIST;
{Octonion number quotient. R and S are octonion numbers, S non-zero. T=R/S. }

procedure ORAND (NL: LIST): LIST;
{Octonion number, random. n is a positive beta-integer. Random rational numbers A and B are generated using RNRAND(n). Then R is the octonion number with real part A and imaginary part B. }

procedure ONREAD (): LIST;
{Octonion number read. The octonion number R is read from the input stream. Any preceding blanks are skipped. }

procedure OSUM (R,S: LIST): LIST;
{Octonion number sum. R and S are octonion numbers. T=R+S. }

procedure ONWRITE (R: LIST);
{Octonion number write. R is a octonion number. R is converted to decimal and written in the output stream. }

6.7 MAS Quaternion Number

procedure QABS (R: LIST): LIST;
{Quaternion number absolute value. R is a quaternion number. S is the absolute value of R, a rational number. }

procedure QCON (R: LIST): LIST;
{Quaternion number conjugate. R is a quaternion number. S is the quaternion conjugate of R. }

procedure QCOMP (R,S: LIST): LIST;
{Quaternion number comparison. R and S are quaternion numbers. t=0 if R=S, t=1 else. }

procedure QDIF (R,S: LIST): LIST;
{Quaternion number difference. R and S are quaternion numbers. T=R-S. }

procedure QDREAD (): LIST;
{Quaternion number decimal read. The quaternion number R is read from the input stream. Any preceding blanks are skipped. }

procedure QDWRITE (R,NL: LIST);
{Quaternion number decimal write. R is a quaternion number. n is a non-negative integer. R is approximated by a decimal fraction D with n decimal digits following the decimal point and D is written in the output stream. The inaccuracy of the approximation is at most (1/2)*10**-n. }

procedure QEXP (A,NL: LIST): LIST;
{Quaternion number exponentiation. A is a quaternion number, n is a non-negative beta-integer. B=A**n. }

procedure QIMi (R: LIST): LIST;
{Quaternion number imaginary part i. R is a quaternion number. b is the imaginary part i of R, a rational number. }

procedure QIMj (R: LIST): LIST;
{Quaternion number imaginary part j. R is a quaternion number. b is the imaginary part j of R, a rational number. }

procedure QIMk (R: LIST): LIST;
{Quaternion number imaginary part k. R is a quaternion number. b is the imaginary part k of R, a rational number. }

procedure QINT (A: LIST): LIST;
{Quaternion number from integer. A is an integer. R is the quaternion number with real part A/1 and imaginary part 0. }

procedure QRE (R: LIST): LIST;
{Quaternion number real part. R is a quaternion number. b is the real part of R, a rational number. }

procedure QRN (A: LIST): LIST;
{Quaternion number from rational number. A is a rational number. R is the quaternion number with real part A and imaginary part 0. }

procedure QRN4 (A, B, C, D: LIST): LIST;
{Quaternion number from 4-tuple of rational numbers. A, B, C and D are rational numbers. R is the quaternion number with real part A and imaginary part i B, imaginary part j C, imaginary part k D. }

procedure QINV (R: LIST): LIST;
{Quaternion number inverse. R is a non-zero quaternion number. S R=1. }

procedure QNEG (R: LIST): LIST;
{Quaternion number negative. R is a quaternion number. S=-R. }

procedure QONE (R: LIST): LIST;
{Quaternion number one. R is a quaternion number. s=1 if R=1, s=0 else. }

procedure QPROD (R,S: LIST): LIST;
{Quaternion number product. R and S are quaternion numbers. T=R*S. }

procedure QQ (R,S: LIST): LIST;
{Quaternion number quotient. R and S are quaternion numbers, S non-zero. T=R/S. }

procedure QRAND (NL: LIST): LIST;
{Quaternion number, random. n is a positive beta-integer. Random rational numbers A and B are generated using RNRAND(n). Then R is the quaternion number with real part A and imaginary part B. }

procedure QNREAD (): LIST;
{Quaternion number read. The quaternion number R is read from the input stream. Any preceding blanks are skipped. }

procedure QSUM (R,S: LIST): LIST;
{Quaternion number sum. R and S are quaternion numbers. T=R+S. }

procedure QNWRITE (R: LIST);
{Quaternion number write. R is a quaternion number. R is converted to decimal and written in the output stream. }

6.8 MAS Rational Number

procedure RNDRD (): LIST;
{Rational number decimal read. The rational number R is read from the input stream. Any preceding blanks are skipped.}

procedure RNDWR (R,NL: LIST);
{Rational number decimal write. R is a rational number. n is a non-negative integer. R is approximated by a decimal fraction D with n decimal digits following the decimal point and D is written in the output stream. The inaccuracy of the approximation is at most (1/2)*10**-n. }

procedure RNDWRS (A,S: LIST);
{Rational number decimal write special. Call RNDWR. }

procedure RNEXP (A,NL: LIST): LIST;
{Rational number exponentiation. A is a rational number, n is a non-negative beta-integer. B=A**n.}

procedure RNMAX (AL,BL: LIST): LIST;
{Rational number maximum. a and b are rational numbers. c is the maximum of a and b.}

procedure RNONE (R: LIST): LIST;
{Rational number one. R is a rational number. s=1 if R=1, s=0 else. }

6.9 MAS Set

procedure SetAdd (e,S:LIST):LIST;
{Set add. S is a set, e is an element. A set containing all elements of S and the element e is returned. If e is not a element of the set S, e is the first element in the list representing S. }

procedure SetAddQ (e,S:LIST):LIST;
{Set add equal. S is a set, e is an element. A set containing all elements of S and the element e is returned. If e is not a element of the set S, e is the first element in the list representing S. }

procedure SetUnion (S1,S2:LIST):LIST;
{Set union. S1 and S2 are sets. The union of S1 and S2 is returned. The elements of S2 not occuring in set1 are added to S1. }

procedure SetUnionQ (S1,S2:LIST):LIST;
{Set union equal. S1 and S2 are sets. The union of S1 and S2 is returned. The elements of S2 not occuring in set1 are added to S1. }

procedure SetElementP (e,S:LIST):BOOLEAN;
{Set element predicate. e is an element, S is a set. SetElementP returns true iff e is a element of S }

procedure SetElementPQ (e,S:LIST):BOOLEAN;
{Set element predicate equal. e is an element, S is a set. SetElementP returns true iff e is a element of S }

procedure SetMinus (e,S:LIST):LIST;
{Set minus. e is an element. S is a set. If e is an element of the set S then a set containing all elements of S except of the element e is returned. Otherwise the set S is returned. S is modified to build the result. }

procedure SetMinusQ (e,S:LIST):LIST;
{Set minus equal. e is an element. S is a set. If e is an element of the set S then a set containing all elements of S except of the element e is returned. Otherwise the set S is returned. S is modified to build the result. }

procedure SetMinusC (e,S:LIST):LIST;
{Set minus constructive. e is an element. S is a set. If e is an element of the set S then a set containing all elements of S except of the element e is returned. Otherwise the set S is returned. }

procedure SetMinusCQ (e,S:LIST):LIST;
{Set minus constructive equal. e is an element. S is a set. If e is an element of the set S then a set containing all elements of S except of the element e is returned. Otherwise the set S is returned. }

procedure SetComplement (S1,S2:LIST):LIST;
{Set complement. S1 and S2 are sets. The complement of S1 with respect to S2 is returned. }

procedure SetComplementQ (S1,S2:LIST):LIST;
{set complement. S1 and S2 are sets. The complement of S1 with respect to S2 is returned. }

6.10 SAC Combinatorical System

procedure ASSPR (A: LIST; VAR PL,ML: LIST);
{Assignment problem. A is a square matrix of beta-integers, say n by n. p is an n-permutation for which the sum on i of A(i,p(i)) is maximal, and m is this maximal sum. All matrix elements A(i,j) must be less than beta in absolute value.}

procedure CSFPAR (L: LIST): LIST;
{Characteristic set from partition. L is a list of non-negative beta- integers (l sub 1, ...,l sub n). C is a characteristic set, with j belonging to C if and only if there is a subset I of the integers from 1 to n such that the sum of the l sub i with i in I=j.}

procedure CSINT (A,B: LIST): LIST;
{Characteristic set intersection. A and B are characteristic sets. C is the intersection of A and B.}

procedure CSSUB (A,B: LIST): LIST;
{Characteristic set subset. A and B are characteristic sets. t=1 if A is a subset of B and otherwise t=0.}

procedure CSUN (A,B: LIST): LIST;
{Characteristic set union. A and B are characteristic sets. C is the union of A and B.}

procedure DAND (AL,BL: LIST): LIST;
{Digit and. a and b are non-negative beta-digits. c is the bit-wise and of a and b.}

procedure DNIMP (AL,BL: LIST): LIST;
{Digit non-implication. a and b are non-negative beta-digits. c is the bit-wise non-implication of a and b.}

procedure DNOT (AL: LIST): LIST;
{Digit not. a is a non-negative beta-digit. b is the bit-wise not of a.}

procedure DOR (AL,BL: LIST): LIST;
{Digit or. a and b are non-negative beta-digits. c is the bit-wise or of a and b.}

procedure IBCIND (A,NL,KL: LIST): LIST;
{Integer binomial coefficient induction. n and k are beta-integers with 0 less than or equal to k less than or equal to n. A is the binomial coefficient n over k. B is the binomial coefficient n over k+1.}

procedure IBCOEF (NL,KL: LIST): LIST;
{Integer binomial coefficient. n and k are beta-integers with 0 less than or equal to k less than or equal to n. A is the binomial coefficient n over k.}

procedure IBCPS (NL,KL: LIST): LIST;
{Integer binomial coefficient partial sum. n and k are beta integers, 0 le k le n. A is the sum on i, from 0 to k, of the binomial coefficient n over i.}

procedure IFACTL (NL: LIST): LIST;
{Integer factorial. n is a non-negative beta-integer. A is n factorial.}

procedure LEXNEX (A: LIST): LIST;
{Lexicographically next. A is a non-null list (a sub 1, ...,a sub m) such that a sub i is a non-null reductant of a sub i+1 for each 1 le i lt m. B is the lexicographically next such list of the same length, if one exists, and is () otherwise.}

procedure LPERM (L,P: LIST): LIST;
{List permute. L is a list (a sub 1, ...,a sub n). P is a list (p sub 1, ...,p sub n) of integers in the range 1, ...,n. LP is the list (a sub p sub 1, ...,a sub p sub n).}

procedure PARTN (NL,P: LIST): LIST;
{Partition, next. n is a positive beta-integer. P is a partition of n. Q is the next partition of n after P in lexicographical order, if any. Otherwise Q=().}

procedure PARTR (NL: LIST): LIST;
{Partition, random. n is a positive beta-integer, n less than or equal to 100. P is a partition of n whose elements are the cycle lengths of a random n-permutation.}

procedure PARTSS (PL: LIST): LIST;
{Partition sumset. p is a partition. A is the sum set of p, a characteristic set.}

procedure PERMR (NL: LIST): LIST;
{Permutation, random. n is a positive integer, n le 100. L is a list of the first n positive integers in random order.}

procedure SDR (S: LIST; VAR A,I: LIST);
{System of distinct representatives. S is a list (s(1), ...,s(n)), n ge 1, where each s(i) is a set of beta-integers represented as a list. Either A is a list (a(1), ...,a(n)) of distinct representatives for (s(1), ...,s(n)) and I=(), or else A=() and I=(i(1), ...,i(k)) is a subsequence of (1, ...,n) such that (s(i(1)), ...,s(i(k))) has no system of distinct representatives.}

procedure SFCS (A: LIST): LIST;
{Set from characteristic set. A is a characteristic set. B is the same set represented as an increasing list of beta-integers.}

6.11 SAC Digit

procedure BITRAN (): LIST;
{Bit, random. b is a random bit, 0 or 1.}

procedure DEGCD (AL,BL: LIST; VAR CL,UL,VL: LIST);
{Digit extended greatest common divisor. a and b are beta-integers, a ge b ge 0. c=GCD(a,b), a beta-integer. a*u+b*v=c, with ABS(u) le b/2c, ABS(v) le a/2c.}

procedure DGCD (AL,BL: LIST): LIST;
{Digit greatest common divisor. a and b are beta-integers, a ge b ge 0. c=GCD(a,b).}

procedure DLOG2 (AL: LIST): LIST;
{Digit logarithm, base 2. a is a beta-digit. If a=0 then n=0. otherwise n=FLOOR(LOG2(ABS(a)))+1.}

procedure DPCC (AL1,AL2: LIST; VAR UL,ULP,VL,VLP: LIST);
{Digit partial cosequence calculation. a1 and a2 are beta-integers, a1 ge a2 gt 0. u, up, v and vp are the last cosequence elements of a1 and a2 which can be guaranteed to correspond to correct quotient digits.}

procedure DPR (AL,BL: LIST; VAR CL,DL: LIST);
{Digit product. a and b are beta-digits. c and d are the unique beta-digits such that a*b=c*beta+d and c*d ge 0.}

procedure DQR (AL1,AL0,BL: LIST; VAR QL,RL: LIST);
{Digit quotient and remainder. a1, a0 and b are beta-integers with a1*a0 ge 0 and ABS(b) gt ABS(a1). q is the integral part of (a1*beta+a0)/b and r is (a1*beta+a0)-b*q. q and r are beta-integers.}

procedure DRAN (): LIST;
{Digit, random. a is a random beta-digit.}

procedure DRANN (): LIST;
{Digit, random non-negative. a is a random non-negative beta-digit. Caution, the low-order bits of a are not very random.}

procedure DSQRTF (AL: LIST; VAR BL,TL: LIST);
{Digit square root function. a is a non-negative beta-integer. b is the floor function of the square root of a and t is the sign of a-b*b.}

6.12 SAC Integer

procedure AADV (L: LIST; VAR AL,LP: LIST);
{Arithmetic advance. L is a list. If L ne () then a=FIRST(L) and LP=RED(L). Otherwise a=0 and LP=().}

procedure IABSF (A: LIST): LIST;
{Integer absolute value function. A is an integer. B=ABS(A).}

procedure ICOMP (A,B: LIST): LIST;
{Integer comparison. A and B are integers. s=SIGN(A-B).}

procedure IDEGCD (AL,BL: LIST; VAR CL,UL1,VL1,UL2,VL2: LIST);
{Integer doubly extended greatest common divisor algorithm. a and b are integers. c=GCD(a,b). a*u1+b*v1=c and a*u2+b*v2=0. If a ne 0 and b ne 0 then ABS(u1) le ABS(b)/(2*c), ABS(v1) le ABS(a)/(2*c), u2=-b/c and v2=a/c. Otherwise u1=v2=SIGN(a), v1=SIGN(b) and u2=-SIGN(b).}

procedure IDIF (A,B: LIST): LIST;
{Integer difference. A and B are integers. C=A-B.}

procedure IDIPR2 (A,B,AL,BL: LIST): LIST;
{Integer digit inner product, length 2. A and B are integers. a and b are beta-integers. C=A*a+B*b.}

procedure IDPR (A,BL: LIST): LIST;
{Integer-digit product. A is an integer. b is a beta-digit. C=A*b.}

procedure IDP2 (A,KL: LIST): LIST;
{Integer division by power of 2. A is an integer. k is a non-negative beta-digit. B is the integral part of A/2**k.}

procedure IDQ (A,BL: LIST): LIST;
{Integer-digit quotient. A is an integer. b is a non-zero beta-digit. C=INTEGER(A/b).}

procedure IDQR (A,BL: LIST; VAR Q,RL: LIST);
{Integer-digit quotient and remainder. A is an integer. b is a non-zero beta-digit. Q is the integral part of A/b and r=A-b*Q.}

procedure IDREM (A,BL: LIST): LIST;
{Integer-digit remainder. A is an integer. b is a non-zero beta-digit. r=A-b*INTEGER(A/b).}

procedure IEGCD (AL,BL: LIST; VAR CL,UL1,VL1: LIST);
{Integer extended greatest common divisor algorithm. a and b are integers. c=GCD(a,b). a*u1+b*v1=c. If a ne 0 and b ne 0 then ABS(u1) le ABS(b)/(2*c) and ABS(v1) le ABS(a)/(2*c). Otherwise u1=SIGN(a) and v1=SIGN(b).}

procedure IEVEN (A: LIST): BOOLEAN;
{Integer even. A is an integer. If A is even then true is returned and otherwise false. }

procedure IEXP (A,NL: LIST): LIST;
{Integer exponentiation. A is an integer. n is a non-negative beta-integer. B=A**n.}

procedure IFCL2 (AL: LIST; VAR ML,NL: LIST);
{Integer, floor and ceiling, logarithm, base 2. a is a non-zero integer. m and n, gamma-integers, are the floor and ceiling of LOG2(ABS(a)) respectively.}

procedure IGCD (A,B: LIST): LIST;
{Integer greatest common divisor. A and B are integers. C=GCD(A,B).}

procedure IGCDCF (A,B: LIST; VAR C,AB,BB: LIST);
{Integer greatest common divisor and cofactors. A and B are integers. C eq GCD(A,B). If C eq 0 then AB eq BB eq 0 and otherwise AB eq A/C,BB eq B/C.}

procedure IHEGCD (A,B: LIST; VAR C,V: LIST);
{Integer half-extended greatest common divisor. A and B are integers. C=GCD(A,B). If A ne 0, B*V=C(mod A), with ABS(V) le ABS(A)/2C. If A=0,V=SIGN(B).}

procedure ILCM (A,B: LIST): LIST;
{Integer least common multiple. A and B are integers. C=LCM(A,B), a nonnegative integer.}

procedure ILCOMB (A,B,UL,VL: LIST): LIST;
{Integer linear combination. A and B are non-negative integers. u and v are beta-integers such that A*u+B*v ge 0. C=A*u+B*v.}

procedure ILOG2 (A: LIST): LIST;
{Integer logarithm, base 2. A is an integer. If A=0 then n=0. Otherwise n=FLOOR(LOG2(ABS(A)))+1, a beta-integer.}

procedure ILWRIT (L: LIST);
{Integer list write. L is a list of integers. The list L is written in the output stream.}

procedure IMAX (AL,BL: LIST): LIST;
{Integer maximum. a and b are integers. c is the maximum of a and b.}

procedure IMIN (AL,BL: LIST): LIST;
{Integer minimum. a and b are integers. c is the minimum of a and b.}

procedure IMP2 (A,HL: LIST): LIST;
{Integer multiplication by power of 2. A is an integer. h is a non-negative beta-integer. B=A*(2**h).}

procedure INEG (A: LIST): LIST;
{Integer negation. A is an integer. B=-A.}

procedure IODD (A: LIST): BOOLEAN;
{Integer odd. A is an integer. If a is odd then true is returned and otherwise false. }

procedure IORD2 (AL: LIST): LIST;
{Integer, order of 2. a is a non-zero integer. n is the largest integer such that 2**n divides a.}

procedure IPOWER (A,L: LIST; VAR B,NL: LIST);
{Integer power. A, greater than or equal to 3, is an odd positive integer. L is a list (p(1),p(2), ...,p(k)) of the first k prime numbers, with p(k) greater than or equal to the base 3 logarithm of A. If A=B**m for some m greater than or equal to 2 then n is the least such m and B=A**(1/n). Otherwise B=0 and n=0.}

procedure IPROD (A,B: LIST): LIST;
{Integer product. A and B are integers. C=A*B.}

procedure IPRODK (A,B: LIST): LIST;
{Integer product, Karatsuba algorithm. A and B are integers. C=A*B.}

procedure IQ (A,B: LIST): LIST;
{Integer quotient. A and B are integers, B ne 0. C is the integral part of A/B.}

procedure IQR (A,B: LIST; VAR Q,R: LIST);
{Integer quotient and remainder. A and B are integers, B ne 0. Q is the quotient, integral part of A/B, and R is the remainder A-B*Q.}

procedure IRAND (NL: LIST): LIST;
{Integer, random. n is a positive beta-integer. A is an integer with random sign and random absolute value less than 2**n.}

procedure IREAD (): LIST;
{Integer read. The integer A is read from the input stream. Any preceding blanks are skipped.}

procedure IREM (A,B: LIST): LIST;
{Integer remainder. A and B are integers, B non-zero. C is the remainder of A and B.}

procedure IROOT (A,NL: LIST; VAR B,TL: LIST);
{Integer root. A is a positive integer. n, greater than or equal to 2, is a beta-integer. B=FLOOR(A**(1/n)) and t=SIGN(A-B**n).}

procedure ISEG (A,NL: LIST; VAR A1,A0: LIST);
{Integer segmentation. A is an integer. n is a positive beta- integer. A1 is the integral part of A/beta**n. A0=A-A1*beta**n.}

procedure ISIGNF (A: LIST): LIST;
{Integer sign function. A is an integer. s=SIGN(A).}

procedure ISQRT (A: LIST; VAR B,TL: LIST);
{Integer square root. A is a non-negative integer. B is the floor function of the square root of A and t is the sign of A-B*B.}

procedure ISSUM (NL,L: LIST): LIST;
{Integer shifted sum. n is a positive integer. L is a list (c(0),c(1), ...,c(k)), k non-negative, of integers c(i) with ABS(c(i)) less than beta**(2*n+1). Either each c(i) is non-negative or each c(i) is non-positive. C is the sum on i, from 0 to k, of c(i)*(beta**(i*n)).}

procedure ISUM (A,B: LIST): LIST;
{Integer sum. A and B are integers. C=A+B.}

procedure ITRUNC (A,NL: LIST): LIST;
{Integer truncation. A is an integer. n is a beta-integer. B=INTEGER(A/2**n).}

procedure IWRITE (A: LIST);
{Integer write. The input integer A is converted to decimal and written in the output stream.}

6.13 SAC Modular Digit and Integer

procedure MDCRA (ML1,ML2,MLP1,AL1,AL2: LIST): LIST;
{Modular digit chinese remainder algorithm. m1 and m2 are positive beta-integers, with GCD(m1,m2)=1 and m=m1*m2 less than beta. mp1 is the inverse of m1 in Z(m2). a1 and a2 are elements of Z(m1) and Z(m2) respectively. a is the unique element of Z(m) such that a is congruent to a1 modulo m1 and a is congruent to a2 modulo m2.}

procedure MDDIF (ML,AL,BL: LIST): LIST;
{Modular digit difference. m is a positive beta-integer. a and b belong to Z sub m. c=a-b.}

procedure MDEXP (ML,AL,NL: LIST): LIST;
{Modular digit exponentiation. m is a positive beta-integer. a belongs to Z sub m. n is a non-negative beta-integer. b=a**n.}

procedure MDHOM (ML,A: LIST): LIST;
{Modular digit homomorphism. m is a positive beta-integer. A is an integer. b is the image of A under the homomorphism H sub m.}

procedure MDINV (ML,AL: LIST): LIST;
{Modular digit inverse. m is a positive beta-integer. a is a unit of Z sub m. b=a**-1.}

procedure MDLCRA (ML1,ML2,L1,L2: LIST): LIST;
{Modular digit list chinese remainder algorithm. m1 and m2 are positive beta-integers, with GCD(m1,m2)=1 and m=m1*m2 less than beta. L1 and L2 are lists of elements of Z(m1) and Z(m2) respectively. L is a list of all a in Z(m) such that a is congruent to a1 modulo m1 and a is congruent to a2 modulo m2 with a1 in L1 and a2 in L2.}

procedure MDNEG (ML,AL: LIST): LIST;
{Modular digit negative. m is a positive beta-integer. a belongs to Z sub m. b=-a.}

procedure MDPROD (ML,AL,BL: LIST): LIST;
{Modular digit product. m is a positive beta-integer. a and b belong to Z sub m. c=a*b.}

procedure MDQ (ML,AL,BL: LIST): LIST;
{Modular digit quotient. m is a positive beta-integer. a and b belong to Z sub m. b is a unit. c=a/b.}

procedure MDRAN (ML: LIST): LIST;
{Modular digit, random. m is a positive beta-digit. a is a random element of Z(m).}

procedure MDSUM (ML,AL,BL: LIST): LIST;
{Modular digit sum. m is a positive beta-integer. a and b belong to Z sub m. c=a+b.}

procedure MIDCRA (M,ML,MLP,A,AL: LIST): LIST;
{Modular integer digit chinese remainder algorithm. M is a positive integer. m is an odd positive beta-integer. GCD(M,m)=1. mp is the inverse of the image of M under the homomorphism H sub m. A and a are elements of Z prime sub M and Z sub m respectively. AS is the unique element of Z prime sub MS which is congruent to A modulo M and congruent to a modulo m, where MS=M*m.}

procedure MIDIF (M,A,B: LIST): LIST;
{Modular integer difference. M is a positive integer. A and B belong to Z sub M. C=A-B.}

procedure MIEXP (M,A,N: LIST): LIST;
{Modular integer exponentiation. M is a positive integer. A is an element of Z(M). N is a non-negative integer. B=A**N in Z(M).}

procedure MIHOM (M,A: LIST): LIST;
{Modular integer homomorphism. M is a positive integer. A is an integer. AS=H sub M(A).}

procedure MIINV (M,A: LIST): LIST;
{Modular integer inverse. M is a positive integer. A is a unit of Z sub M. B=A**-1.}

procedure MINEG (M,A: LIST): LIST;
{Modular integer negation. M is a positive integer. A belongs to Z sub M. B=-A.}

procedure MIPROD (M,A,B: LIST): LIST;
{Modular integer product. M is a positive integer. A and B belong to Z(M). C=A*B in Z(M).}

procedure MIQ (M,A,B: LIST): LIST;
{Modular integer quotient. M is a positive integer. A and B belong to Z sub M. B is a unit. C=A/B.}

procedure MIRAN (M: LIST): LIST;
{Modular integer, random. M is a positive integer. R is a uniformly distributed random element

of Z sub M.}

procedure MISUM (M,A,B: LIST): LIST;
{Modular integer sum. M is a positive integer. A and B belong to Z sub M. C=A+B.}

procedure SMFMI (M,A: LIST): LIST;
{Symmetric modular from modular integer. M is a positive integer. A belongs to Z sub M. B belongs to Z prime sub M with B=A(modulo M).}

6.14 SAC Factorization and Prime Number

procedure DPGEN (ML, K: LIST): LIST;
{Digit prime generator. K and m are positive beta-integers. L is the list (p(1),...,p(r)) of all prime numbers p such that m le p lt m+2*K, with p(1) lt p(2) lt ... lt p(r). A local array is used.}

procedure FRESL (NL: LIST; VAR ML,L: LIST);
{Fermat residue list. n is a positive integer with no prime divisors less than 17. m is a positive beta-integer and L is an ordered list of the elements of Z(m) such that if x**2-n is a square then x is congruent to a (modulo m) for some a in L.}

procedure FRLSM (ML,AL: LIST): LIST;
{Fermat residue list, single modulus. m is a positive beta-integer. a belongs to Z(m). L is a list of the distinct b in Z(m) such that b**2-a is a square in Z(m).}

procedure GDPGEN (ML: LIST; KL: INTEGER): LIST;
{Gaussian digit prime generator. m and k are positive beta-integers. L is the list (p(1),...,p(r)) of all prime numbers p such that m is less than or equal to p, p is less than m+4*k and p is congruent to 3 mod 4, with p(1) lt p(2) lt ... lt p(r). A local array is used.}

procedure IFACT (NL: LIST): LIST;
{Integer factorization. n is a positive integer. F is a list (q(1), q(2),...,q(h)) of the prime factors of n, q(1) le q(2) le ... le q(h), with n equal to the product of the q(i).}

procedure ILPDS (NL,AL,BL: LIST; VAR PL,NLP: LIST);
{Integer large prime divisor search. n is a positive integer with no prime divisors less than 17. 1 le a le b le n. A search is made for a divisor p of the integer n, with a le p le b. If such a p is found then np=n/p, otherwise p=1 and np=n. A modular version of fermats method is used, and the search goes from a to b.}

procedure IMPDS (NL,AL,BL: LIST; VAR PL,QL: LIST);
{Integer medium prime divisor search. n, a and b are positive integers such that a le b le n and n has no positive divisors less than a. If n has a prime divisor in the closed interval from a to b then p is the least such prime and q=n/p. Otherwise p=1 and q=n.}

procedure ISPD (NL: LIST; VAR F,ML: LIST);
{Integer small prime divisors. n is a positive integer. F is a list of primes (q(1),q(2),...,q(h)), h non-negative, q(1) le q(2) le ... lt q(h), such that n is equal to m times the product of the q(i) and m is not divisible by any prime in SMPRM. Either m=1 or m gt 1,000,000.}

procedure ISPT (ML,MLP,F: LIST): LIST;
{Integer selfridge primality test. m is an integer greater than or equal to 3. mp=m-1. F is a list (q(1),q(2),...,q(k)), q(1) le q(2) le ... le q(k), of the prime factors of mp, with mp equal to the product of the q(i). An attempt is made to find a root of unity modulo m of order m-1. If the existence of such a root is discovered then m is prime and s=1. If it is discovered that no such root exists then m is not a prime and s=-1. Otherwise the primality of m remains uncertain and s=0.}

6.15 SAC Rational Number

procedure RIRNP (I,CL: LIST): LIST;
{Rational interval rational number product. I is an interval with rational endpoints. c is a rational number. J is the interval I*c.}

procedure RNABS (R: LIST): LIST;
{Rational number absolute value. R is a rational number. S is the absolute value of R.}

procedure RNCEIL (RL: LIST): LIST;
{Rational number, ceiling of. r is a rational number. a=CEILING(r), an integer.}

procedure RNCOMP (R,S: LIST): LIST;
{Rational number comparison. R and S are rational numbers. t=SIGN(R-S).}

procedure RNDEN (R: LIST): LIST;
{Rational number denominator. R is a rational number. b is the denominator of R, a positive integer.}

procedure RNDIF (R,S: LIST): LIST;
{Rational number difference. R and S are rational numbers. T=R-S.}

procedure RNDWR (R,NL: LIST);
{Rational number decimal write. R is a rational number. n is a non-negative integer. R is approximated by a decimal fraction D with n decimal digits following the decimal point and D is written in the output stream. The inaccuracy of the approximation is at most (1/2)*10**-n. If ABS(D) is greater than ABS(R) then the last digit is followed by a minus sign, if ABS(D) is less than ABS(R) then by a plus sign.}

procedure RNFCL2 (AL: LIST; VAR ML,NL: LIST);
{Rational number floor and ceiling of logarithm, base 2. a is a non- zero rational number. m=FLOOR(LOG2(ABS(a))) and n=CEILING(LOG2(ABS(a))) are gamma-integers.}

procedure RNFLOR (RL: LIST): LIST;
{Rational number, floor of. r is a rational number. a=FLOOR(r), an integer.}

procedure RNINT (A: LIST): LIST;
{Rational number from integer. A is an integer. R is the rational number A/1.}

procedure RNINV (R: LIST): LIST;
{Rational number inverse. R is a non-zero rational number. S=1/R.}

procedure RNNEG (R: LIST): LIST;
{Rational number negative. R is a rational number. S=-R.}

procedure RNNUM (R: LIST): LIST;
{Rational number numerator. R is a rational number. a is the numerator of R, an integer.}

procedure RNPROD (R,S: LIST): LIST;
{Rational number product. R and S are rational numbers. T=R*S.}

procedure RNP2 (KL: LIST): LIST;
{Rational number power of 2. k is a gamma-integer. r=2**k, a rational number.}

procedure RNQ (R,S: LIST): LIST;
{Rational number quotient. R and S are rational numbers, S non-zero. T=R/S.}

procedure RNRAND (NL: LIST): LIST;
{Rational number, random. n is a positive beta-integer. Random integers A and B are generated using IRAND(n). Then R=A/(ABS(B)+1), reduced to lowest terms.}

procedure RNREAD (): LIST;
{Rational number read. The rational number R is read from the input stream. Any preceding blanks are skipped.}

procedure RNRED (A,B: LIST): LIST;
{Rational number reduction to lowest terms. A and B are integers, B non-zero. R is the rational number A/B in canonical form.}

procedure RNSIGN (R: LIST): LIST;
{Rational number sign. R is a rational number. s=SIGN(R).}

procedure RNSUM (R,S: LIST): LIST;
{Rational number sum. R and S are rational numbers. T=R+S.}

procedure RNWRIT (R: LIST);
{Rational number write. R is a rational number. R is converted to decimal and written in the output stream.}

6.16 SAC Set

procedure LBIBMS (L: LIST): LIST;
{List of beta-integers bubble-merge sort. L is an arbitrary list of beta-integers, possibly with repetitions. M is the result of sorting L into non-decreasing order. A combination of bubble-sort and merge- sort is used. The list L is modified to produce M.}

procedure LBIBS (L: LIST);
{List of beta-integers bubble sort. L is an arbitrary list of beta-integers, with possible repetitions. L is sorted into non-decreasing order by the bubble-sort method. The list L, though not its location, is modified.}

procedure LBIM (L1,L2: LIST): LIST;
{List of beta-integers merge. L1 and L2 are arbitrary lists of beta-integers in non-decreasing order. L is the merge of L1 and L2. L1 and L2 are modified to produce L.}

procedure SCOMP (AL,L: LIST): LIST;
{Set composition. a is a beta-integer, L is a set of beta-integers. LP is the union of SET(a) and L. }

procedure SDIFF (A,B: LIST): LIST;
{Set difference. A and B are sets of beta-integers. C=A-B.}

procedure SINTER (A,B: LIST): LIST;
{Set intersection. A and B are sets of beta-integers. C is the intersection of A and B.}

procedure SUNION (A,B: LIST): LIST;
{Set union. A and B are sets of beta-integers. C is the union of A and B.}

procedure USCOMP (AL,L: LIST): LIST;
{Unordered set composition. a is an object, L is an unordered set. LP is the union of SET(a) and L. }

procedure USDIFF (A,B: LIST): LIST;
{Unordered set difference. A and B are unordered sets. C is the difference A-B.}

procedure USINT (A,B: LIST): LIST;
{Unordered set intersection. A and B are unordered sets. C is the intersection of A and B.}

procedure USUN (A,B: LIST): LIST;
{Unordered set union. A and B are unordered sets. C is the union of A and B.}

Chapter 7

Polynomial Systems Algorithms

7.1 DIP Common Polynomial System

procedure BACKUB ();
{Backspace until blank. }

procedure CLIN (): LIST;
{Character list in. If a character list is next in the input stream then it is read, else L is empty. }

procedure DILBSO (A: LIST);
{Distributive polynomial list bubble sort. A is a list of lists of base coefficients and exponent vectors. Each element of A is sorted with respect to the termordering defined in EVORD by the bubble-sort method, two monomials with equal exponents will lead to an error. The lists in A but not there location, are modified.}

procedure DILFPL (RL,A: LIST): LIST;
{Distributive polynomial list from polynom list. A is a list of polynomials in r variables, r ge 0. Every polynomial in A is converted to distributive representation and returned in B. }

procedure DILPERM (dil,perm: LIST):LIST;
{distributive polynomial list permutation of variables. The variable dil is a list of distributive polynomials in r variables, perm is a permutation. In each distributive polynomial of the list dil the variables are permuted with respect to perm. }

procedure DIPADM (A: LIST; VAR EL,FL,BL,B: LIST);
{Distributive polynomial advance main variable. A is a distributive polynomial in one or more variables. e is the degree of A, b is the leading coefficient of A, B is the reductum of A, f is the degree of B.}

procedure DIPADS (A,IL,SL: LIST; VAR EL,FL,BL,B: LIST);
{Distributive polynomial advance and substitute. A is a distributive polynomial, i is the specified variable, 1 le i le r=DIPNOV(A), s is the new exponent of b in the i-th variable. e is the exponent of the leading monomial of A in the i-th variable, let bs be part of the coefficient of xi**e then b = bs * xi**s, B = A - bs*xi**e, f is the exponent of the leading monomial of B in the i-th variable.}

procedure DIPADV (A,IL: LIST; VAR EL,FL,BL,B: LIST);
{Distributive polynomial advance. A is a distributive polynomial, i is the specified variable, 1 le i le r=DIPNOV(A). e is the exponent of the leading monomial of A in the i-th variable, b is part of the coefficient of xi**e of A, B = A - b*xi**e, f is the exponent of the leading monomial of B in the i-th variable.}

procedure DIPBSO (A: LIST);
{Distributive polynomial bubble sort. A is a list of base coefficients and exponent vectors, A is sorted with respect to the termordering defined in EVORD by the bubble-sort method, two monomials with equal exponents will lead to an error. The list A but not its location, is modified.}

procedure DIPCMP (EL,A: LIST): LIST;
{Distributive polynomial composition. A is a distributive polynomial in r variables. e is an exponent. Let t=r+1, then B(x1, ...,xr,xt)=A(x1, ...,xr)*xt**e.}

procedure DIPDEG (A: LIST): LIST;
{Distributive polynomial degree. A is a distributive polynomial. n is the degree of A in its main variable.}

procedure DIPDPV (A,SL,QL: LIST): LIST;
{Distributive polynomial division by power of variable. A is a distributive polynomial in r variables. s is the desired variable to be divided, s le r. q is a beta-integer. Q = A / (xs**q). }

procedure DIPERM (A,P: LIST): LIST;
{Distributive polynomial permutation of variables. A is a distributive polynomial, in r variables, r ge 0. P is a list (p sub 1, ...,p sub r) whose elements are the beta-digits 1 through r. B(x sub (p sub 1), ...,x sub (p sub r)) =A(x sub 1, ...,x sub r). }

procedure DIPEVL (A: LIST): LIST;
{Distributive polynomial exponent vector leading monomial. A is a distributive polynomial. u is the exponent vector of the leading monomial of A. }

procedure DIPEVP (A,EL: LIST): LIST;
{Distributive polynomial exponent vector product. A is a distributive polynomial, e is an exponent vector C=A*(x**e). }

procedure DIPEXC (A,ILP,JLP: LIST): LIST;
{Distributive polynomial exchange variables. A is a distributive polynomial, the variables ip and jp are exchanged, B=(x1, ...,xip, ...,xjp, ...,xr)=A(x1, ...,xjp, ...,xip, ...,xr), 0 le ip, jp le DIPNOV(A).}

procedure DIPFMO (AL,EL: LIST): LIST;
{Distributive polynomial from monomial. A is a non zero distributive polynomial with a as its leading base coefficient and e as is its exponent vector of the leading monomial. }

procedure DIPFP (RL,A: LIST): LIST;
{Distributive polynomial from polynomial. A is a polynomial in r variables, r ge 0. B is the result of converting A from recursive to distributive representation. Modified version original version by G. E. Collins. }

procedure DIPINV (A,JL,KL: LIST): LIST;
{Distributive polynomial introduction of new variables. A is a distributive polynomial in r variables. k ge 0, 0 le j le r. B(x1, ...,xj,y1, ...,yk,xj+1, ...,xr)=A(x1, ...,xr).}

procedure DIPLBC (A: LIST): LIST;
{Distributive polynomial leading base coefficient. A is a distributive polynomial. a is the leading base coefficient of A.}

procedure DIPLDC (A: LIST): LIST;
{Distributive polynomial leading coefficient. A is a distributive polynomial in one or more variables. a is the leading coefficient of A.}

procedure DIPLM (L1,L2: LIST): LIST;
{Distributive polynomial list merge. L1 and L2 are lists of non zero distributive polynomials in non decreasing order. L is the merge of L1 and L2. L1 and L2 are modified to produce L. }

procedure DIPLPM (A: LIST): LIST;
{Distributive polynomial list pair-merge sort. A is a list of non zero distributive polynomials. B

is the result of sorting A into non-decreasing order. Pairs of polynomials are merged. The list A is modified to produce B. }

procedure DIPLRS (A: LIST);
{Distributive polynomial list re-sort. A is a list of distributive polynomials in r variables, r ge 0. The polynomials in A are re-sorted. }

procedure DIPMAD (A: LIST; VAR AL,EL,AP: LIST);
{Distributive polynomial monomial advance. A is a non zero distributive polynomial. a is its leading base coefficient, e is the exponent vector of the leading monomial of A. AP is the distributive polynomial a without its leading monomial, or the empty list. }

procedure DIPMCP (AL,EL,A: LIST): LIST;
{Distributive polynomial monomial composition. A is an emty list or a non zero distributive polynomial. AP is a non zero distributive polynomial with a as its leading base coefficient, e as is its exponent vector of the leading monomial and A as its monomial reductum. }

procedure DIPMPM (A,PL: LIST): LIST;
{Distributive polynomial multiplication by power of main variable. A is a distributive polynomial in r variables. p is a beta- integer. B = A * (xr**p). }

procedure DIPMPV (A,SL,PL: LIST): LIST;
{Distributive polynomial multiplication by power of variable. A is a distributive polynomial in r variables. s is the specified variable to be multiplicated, 1 le s le r. p is a beta-integer. B = A * (xs**p). }

procedure DIPMRD (A: LIST): LIST;
{Distributive polynomial monomial reductum. A is a distributive polynomial. B is the distributive polynomial a without the leading monomial of A. }

procedure DIPMST (A,AL,EL: LIST);
{Distributive polynomial monomial set. A is a non zero distributive polynomial. Its leading base coefficient is set to a and its exponent vector of the leading monomial is set to e. }

procedure DIPNBC (A: LIST): LIST;
{Distributive polynomial number of base coefficients. A is a distributive polynomial. l is the number of base coefficients.}

procedure DIPNOV (A: LIST): LIST;
{Distributive polynomial number of variables. A is a distributive polynomial. r is the number of variables, r ge 0. If A=0 then r is set to zero. }

procedure DIPRED (A: LIST): LIST;
{Distributive polynomial reductum. A is a distributive polynomial, in one or more variables. B is the reductum of A.}

procedure DIPTBC (A: LIST): LIST;
{Distributive polynomial trailing base coefficient. A is a distributive polynomial. a is the trailing base coefficient.}

procedure DIPTCF (A: LIST): LIST;
{Distributive polynomial trailing coefficient. A is a distributive polynomial. a is the trailing coefficient of A.}

procedure DIPTCS (A,IL: LIST): LIST;
{Distributive polynomial trailing coefficient specified variable. A is a distributive polynomial in r variables. a is the trailing coefficient of A with respect to the i-th variable, 1 le i le r. }

procedure DIPTDG (A: LIST): LIST;
{Distributive polynomial total degree. A is a distributive polynomial. n is the total degree of A.}

procedure DIPUNT (A: LIST): LIST;
{Distributive polynomial univariate test. A is a distributive polynomial. If a is univariate then

t=1, otherwise t=0.}

procedure DIPUV (A: LIST): LIST;
{Distributive polynomial univariate variable output. A is a distributive polynomial. If A is univariate then t=i, otherwise t=0. were i is the index of the variable in which A is univariate. If A is constant then t= -1. }

procedure EPREAD (): LIST;
{Exponent read. If ** is found in the input stream then e=AREAD, else e=1. }

procedure EVCADD (U,IL,EL: LIST; VAR V,FL: LIST);
{Exponent vector component add. U=(u1, ...,ur) is an exponent vector of length r, e is added to the i-th component, 1 le i le r, f=ui+e, V=(u1, ...,ui+e, ...,ur). }

procedure EVCOMP (U,V: LIST): LIST;
{Exponent vector compare. U=(u1, ...,ur), V=(v1, ...vr) are exponent vectors. r is the length of U and V. t=0 if U eq V. t=1 if U gt V. t=-1 if U lt V. eq, gt, lt with respect to the ordering of the exponent vectors specified in the global variable EVORD. Lexicographical, inverse lexicographical, graded lexicograhpical, inverse graded lexicographical orderings are possible. }

procedure EVCSUB (U,IL,EL: LIST; VAR V,FL: LIST);
{Exponent vector component subtract. U=(u1, ...,ur) is an exponent vector of length r, e is subtracted from the i-th component, 1 le i le r, V=(u1, ...,ui-e, ...,ur), f=ui. }

procedure EVDEL (U,IL: LIST; VAR V,EL: LIST);
{Exponent vector delete. U=(u1, ...,ur) is an exponent vector of length r. i is the component to be deleted, 1 le i le r. V=(u1, ...,ui-1,ui+1, ...,ur), e=ui.}

procedure EVDER (U,IL,EL: LIST; VAR V,FL: LIST);
{Exponent vector derivation. U=(u1, ...,ur) is an exponent vector of length r, from the i-th component e-times one is subtracted and f is multiplied with the result. V=(u1, ...,ui-e, ...,ur). If f=0 then V is undefined. }

procedure EVDFSI (U,V: LIST; VAR W,SL: LIST);
{Exponent vector difference and sign. U=(u1, ...,ur), V=(v1, ...,vr) are exponent vectors of length r. W=(w1, ...,wr) is the componentwise difference of U and V. s is the EVSIGN of W. If s=-1 then W is undefined.}

procedure EVDIF (U,V: LIST): LIST;
{Exponent vector difference. U=(u1, ...,ur), V=(v1, ...,vr) are exponent vectors of length r. W=(w1, ...,wr) is the componentwise difference of U and V.}

procedure EVDOV (U: LIST): LIST;
{Exponent vector dependency on variables. U is an exponent vector. V is the list (j1, ...,jn) where each j is the index of a variable with non zero exponent in U. }

procedure EVEXC (U,IL,JL: LIST): LIST;
{Exponent vector exchange. U=(u1, ...,ui, ...,uj, ...,ur) is an exponent vector of length r. The components ui and uj are exchanged, 1 le i lt j le r. V=(u1, ...,uj, ...,ui, ...,ur).}

procedure EVIGLC (U,V: LIST): LIST;
{Exponent vector inverse graded lexicographical compare. U=(u1, ...,ur), V=(v1, ...vr) are exponent vectors. t=0 if U eq V. t=1 if U gt V. t=-1 if U lt V. eq, gt, lt with respect to the inverse graded lexicographical ordering of the exponent vectors. r is the length of U and V.}

procedure EVILCI (U,V: LIST): LIST;
{Exponent vector inverse lexicographical compare inverse exponent vector. U=(u1, ...,ur), V=(v1, ...vr) are exponent vectors. t=0 if U eq V. t=1 if U gt V. t=-1 if U lt V. eq, gt, lt with respect to the inverse lexicographical ordering of the exponent vectors. r is the length of U and V.}

procedure EVILCP (U,V: LIST): LIST;
{Exponent vector inverse lexicographical compare. U=(u1, ...,ur), V=(v1, ...vr) are exponent

vectors. t=0 if U eq V. t=1 if U gt V. t=-1 if U lt V. eq, gt, lt with respect to the inverse lexicographical ordering of the exponent vectors. r is the length of U and V.}

procedure EVITDC (U,V: LIST): LIST;
{Exponent vector inverse total degree compare. U=(u1, ...,ur), V=(v1, ...vr) are exponent vectors. t=0 if U eq V. t=1 if U gt V. t=-1 if U lt V. eq, gt, lt with respect to buchbergers total degree ordering of the exponent vectors. r is the length of U and V.}

procedure EVLFCP (L,U,V: LIST): LIST;
{Exponent vector linear form compare. U=(u1, ...,ur), V=(v1, ...,vr) are exponent vectors of length r. L is an univariate integral polynomial vector. t=0 if U eq V. t=1 if U gt V. t=-1 if U lt V. eq, gt, lt with respect to the ordering of the exponent vectors determined by the linear form.}

procedure EVLCM (U,V: LIST): LIST;
{Exponent vector least common multiple. U=(u1, ...,ur), V=(v1, ...,vr) are exponent vectors of length r. W=(w1, ...,wr) is the least common multiple of U and V. }

procedure EVMT (U,V: LIST): LIST;
{Exponent vector multiple test. U=(u1, ...,ur), V=(v1, ...,vr) are exponent vectors of length r. t=1 if U is a multiple of V, t=0 else. }

procedure EVNNZE (U: LIST): LIST;
{Exponent vector number of non zero exponents. U is an exponent vector. n is the number of non zero exponents of U. }

procedure EVOWRITE (EVO: LIST);
{Exponent vector order write. EVO is an exponent vector order. A description of EVO is written to the output stream. inverse refers to the order of variables (in VALIS). ascending means the inverted order (if x¡y then x¿y wrt. the inverted order). }

procedure EvordWrite ();
{Evord Write. Writes a description of EVORD to the output stream. }

procedure EVRAND (RL,KL: LIST): LIST;
{Exponent vector random. r is the length of U. k is a positive beta-digit such that every component of U will be less than k and k lt beta. U is a random exponent vector.}

procedure EVRASP (RL,KL,QL: LIST): LIST;
{Exponent vector random. r is the length of U. k is a positive beta-digit such that every component of U will be less than k and k lt beta. U is a random exponent vector.}

procedure EVSIGN (U: LIST): LIST;
{Exponent vector signum. U=(u1, ...,ur) is an exponent vector of length r. t=0 if all components are eq 0, t=1 if all components are ge 0, else t=-1.}

procedure EVSU (U,IL,FL: LIST; VAR V,EL: LIST);
{Exponent vector substitution. U=(u1, ...,ui, ...,ur) is an exponent vector of length r. The i-th component is changed into f. 1 le i le r. e=ui. V=(u1, ...,ui-1,f,ui+1, ...,ur). }

procedure EVSUM (U,V: LIST): LIST;
{Exponent vector sum. U=(u1, ...,ur), V=(v1, ...,vr) are exponent vectors of length r. W=(u1+v1, ...,ur+vr) is the componentwise sum of U and V. }

procedure EVTDEG (U: LIST): LIST;
{Exponent vector total degree. U is an exponent vector. n is the sum of the components of U.}

procedure PBCLI (RL,A: LIST): LIST;
{Polynomial base coefficients list. A is a polynomial in r variables. B is the list of the base coefficients of A. }

procedure PFDIP (A: LIST; VAR RL,B: LIST);
{Polynomial from distributive polynomial. A is a distributive polynomial. B is the result of converting A to recursive representation, r is the number of variables of B, r ge 0. Modified

version, original version by G. E. Collins. }

procedure PLFDIL (A: LIST; VAR RL,B: LIST);
{Polynomial list from distributive polynom list. A is a list of distributive polynomials in r variables, r ge 0. Every polynomial in A is converted to recursive representation and stored in B. }

procedure PMPV (RL,A,IL,NL: LIST): LIST;
{Polynomial multiplication by power of variable. A is a polynomial in r variables. 1 le i le r and n is a beta-integer. B=A*(x sub i)**n. }

procedure PPERMV (RL,A,P: LIST): LIST;
{Polynomial permutation of variables. A is a polynomial in r variables, r ge 0. P is a list (p sub 1, ...,p sub r) whose elements are the beta-digits 1 through r. B(x sub (p sub 1), ...,x sub (p sub r))=A(x sub 1, ..., x sub r).}

procedure STVL (RL: LIST): LIST;
{Standard variable list. r is the number of variables. V is the variable list for the variables x1, ...,xr. }

procedure DIP2AD (P,d,rest: LIST): LIST;
{distributive polynomial to arbitrary domain. P is a polynomial in distributive representation, d is a domain number, rest is a domain descriptor, returns P with added domain numbers and descriptors }

procedure AD2DIP (P: LIST): LIST;
{arbitrary domain to distributive polynomial. P is a polynomial in distributive representation with domain numbers and descriptors, returns P without domain numbers and descriptors }

7.2 DIP Integral

procedure DIIFRP (A: LIST): LIST;
{Distributive integral polynomial from rational polynomial. A is a distributive rational polynomial, B is the primitive positive associate integral polynomial of A. }

procedure DIILFR (A: LIST): LIST;
{Distributive integral polynomial list from rational polynomial list. A is a list of distributive rational polynomial, B is a list of primitive positive associate integral polynomials of the polynomials of A. }

procedure DIILFRCD (A: LIST): LIST;
{DIP integral list from DIP rational list using common denominator. A is a list of distributive rational polynomial, B is a list of associate integral polynomials of the polynomials of A. All polynomials in B are multiplied by a least common multiple of all denominators of all polynomials in A. }

procedure DIILRD (V: LIST): LIST;
{Distributive integral polynomial list read. V is a variable list. A list of distributive integral polynomials in r variables, where r=length(V), r ge 0, is read from the input stream. any blanks preceding A are skipped. }

procedure DIILWR (A,V: LIST);
{Distributive integral polynomial list write. V is a variable list. A list of distributive integral polynomials in r variables, where r=length(V), r ge 0, is written to the output stream. }

procedure DIIPAB (A: LIST): LIST;
{Distributive integral polynomial absolute value. A is a distributive integral polynomial. B is the absolute value of A.}

procedure DIIPCP (A: LIST; VAR CL,AP: LIST);
{Distributive integral polynomial content and primitive part. A is an distributive integral polynomial, c is the integer content and AP is the positive primitive part of A. }

procedure DIIPDF (A,B: LIST): LIST;
{Distributive integral polynomial difference. A and B are distributive integral polynomials. C=A-B.}

procedure DIIPDM (A: LIST): LIST;
{Distributive integral polynomial derivation main variable. A is a distributive polynomial. B is the derivation of A with respect to its main variable.}

procedure DIIPDR (A,IL: LIST): LIST;
{Distributive integral polynomial derivation. A is a distributive polynomial. B is the derivation of A with respect to its i-th variable, 0 le i le DIPNOV(A).}

procedure DIIPEM (A,AL: LIST): LIST;
{Distributive integral polynomial evaluation of main variable. A is a distributive integral polynomial. a is an integer. B(x1, ...,x(r-1))=A(x1, ...,x(r-1),a). }

procedure DIIPEV (A,IL,AL: LIST): LIST;
{Distributive integral polynomial evaluation of the i-th variable. A is a distributive integral polynomial, 1 le i le DIPNOV(A), a is an integer. B(x1, ...,x(i-1),x(i+1), ...,xr)= A(x1, ...,x(i-1),a,x(i+1), ...,xr). }

procedure DIIPEX (A,NL: LIST): LIST;
{Distributive integral polynomial exponentiation. A is a distributive integral polynomial, n is a non-negative beta- integer. B=A**n. 0**0 is by definition a polynomial in zero variables. }

procedure DIIPHD (A,IL,NL: LIST): LIST;
{Distributive integral polynomial higher derivation. A is a distributive integral polynomial. B is the n-th derivation of A with respect to its i-th variable, 0 le i le DIPNOV(A). }

procedure DIIPIP (A,BL: LIST): LIST;
{Distributive integral polynomial integer product. A is a distributive integral polynomial, b is an integer. C=A*b.}

procedure DIIPIQ (A,BL: LIST): LIST;
{Distributive integral polynomial integer quotient. A is a distributive integral polynomial, b is a nonzero integer, and b divides any coefficient of A. C=A/b.}

procedure DIIPLS (A: LIST): LIST;
{Distributive integral polynomial list sum. A is a circular list of distributive integral polynomials. B is the sum of all polynomials in A. }

procedure DIIPMN (A: LIST): LIST;
{Distributive integral polynomial maximum norm. A is a distributive integral polynomial. b is the maximum norm of A.}

procedure DIIPNG (A: LIST): LIST;
{Distributive integral polynomial negative. B= -A.}

procedure DIIPON (A: LIST): LIST;
{Distributive integral polynomial one. A is a distributive integral polynomial. If A=1 then t=1, otherwise t=0.}

procedure DIIPPR (A,B: LIST): LIST;
{Distributive integral polynomial product. A and B are distributive integral polynomials. C=A*B.}

procedure DIIPPS (A,B: LIST): LIST;
{Distributive integral polynomial pseudo-remainder. A and B are distributive integral polynomials, B ne 0. C is the pseudo-remainder of A and B. }

procedure DIIPQ (A,B: LIST): LIST;
{Distributive integral polynomial quotient. A and B are distributive integral polynomials. B is a non zero divisor of A. C=B/A.}

procedure DIIPQR (A,B: LIST; VAR Q,R: LIST);
{Distributive integral polynomial quotient and remainder. A and B are distributive integral polynomials with B ne 0. Q and R are unique distributive integral polynomials such that either B divides A, so Q=A/B and R=0 or B does not divide A, so A=B*Q+R with DEG(R) minimal.}

procedure DIIPRA (RL,KL,LL,EL: LIST): LIST;
{Distributive integral polynomial random. k, l and e are positive beta-digits. e is the maximal permitted exponent of A in any variable. A is a random distributive integral polynomial in r variables max norm of a lt 2**k and maximal l base coefficients. }

procedure DIIPRD (V: LIST): LIST;
{Distributive integral polynomial read. V is a variable list. A distributive integral polynomial A in r variables, where r=length(V), r ge 0, is read from the input stream. Any blanks preceding A are skipped. Modified version, orginal version by G. E. Collins. }

procedure DIIPSG (A: LIST): LIST;
{Distributive integral polynomial sign. A is a distributive integral polynomial. s is the sign of the leading base coefficient of A.}

procedure DIIPSM (A,B: LIST): LIST;
{Distributive integral polynomial sum. A and B are distributive integral polynomials. C=A+B. }

procedure DIIPSN (A: LIST): LIST;
{Distributive integral polynomial sum norm. A is a distributive integral polynomial. b is the sum norm of A.}

procedure DIIPSO (A: LIST): LIST;
{Distributive integral polynomial sort. A is a list of integer base coefficients and exponent vectors, A is sorted with respect to the actual term order, two terms with equal exponent vectors are added. }

procedure DIIPSU (A,IL,B: LIST): LIST;
{Distributive integral polynomial substitution. A and B are distributive integral polynomials, 1 le i le r=DIPNOV(A). E(x1, ...,x(i-1),x(i+1), ...,xr)=A(x1, ...,x(i-1), B(x1, ...,x(i-1),x(i+1), ...,xr),x(i+1), ...,xr). }

procedure DIIPSV (A,B: LIST): LIST;
{Distributive integral polynomial substitution for main variable. A and B are distributive integral polynomials. t=DIPNOV(A)-1. C(x1, ...,xt)=A(x1, ...,xt,B(x1, ...,xt)). }

procedure DIIPTM (A,HL: LIST): LIST;
{Distributive integral polynomial translation main variable. A is a distributive integral polynomial, h is an integer. B(x1, ...xr)=A(x1, ...,x(r-1),xr+h). r=DIPNOV(A). }

procedure DIIPTR (A,HL,IL: LIST): LIST;
{Distributive integral polynomial translation. A is a distributive integral polynomial, h is an integer, the i-th variable is translated. 1 le i le r=DIPNOV(A). B(x1, ...,xr)=A(x1, ...,xi+h, ...,xr).}

procedure DIIPWR (A,V: LIST);
{Distributive integral polynomial write. A is a distributive integral poynomial in r variables, r ge 0. V is a variable list for A. A is written in the output stream. Modified version, original version by G. E. Collins. }

procedure DIIPWV (A: LIST);
{Distributive integral polynomial write with standard variable list. A is a distributive integral poynomial. The standard variable list is used. A is written in the output stream.}

procedure DIIRAS (RL,KL,LL,EL,QL: LIST): LIST;
{Distributive integral polynomial random sparse exponent vector. k, l and e are positive beta-digits. e is the maximal permitted exponent of A in any variable. A is a random distributive integral polynomial in r variables max norm of a lt 2**k and maximal l base coefficients. }

7.3 DIP Integer Polynomial

procedure VIPIIP (RL,A,B: LIST): LIST;
{Vector of integral polynomials with vector of integers inner product. A is a vector of integral polynomials in r variables, r non-negative. B is a vector of integers. C is the inner product of A and B.}

procedure HIPRAN (RL,KL,QL,NL: LIST): LIST;
{Homogeneous integral polynomial random. k is a positive beta-digit. q is a rational number q1/q2 with 0 lt q1 le q2 lt beta. n is a non-negative beta-digit r ge 0. A is a random homogeneous integral polynomial in r variables with homogeneous degree n. max norm of A lt 2**k and q is the probability that any particular term of A has a non-zero coefficient.}

procedure IPRAN (RL,KL,QL,N: LIST): LIST;
{Integral polynomial random. k is a positive beta-digit. q is a rational number q1/q2 with 0 lt q1 le q2 lt beta. N is a list (n sub r, ...,n sub 1) of non-negative beta-digits r ge 0. A is a random integral polynomial in r variables with deg sub i of a le n sub i + 1 for 1 le i le r. Max norm of A lt 2**k and q is the probability that any particular term of A has a non-zero coefficient. Modified version, original version by G. E. Collins. }

7.4 DIP Rational

procedure DIRFIP (A: LIST): LIST;
{Distributive rational polynomial from integral polynomial. A is a distributive integral polynomial, B is the monic associate rational polynomial of A. }

procedure DIRLRD (V: LIST): LIST;
{Distributive rational polynomial list read. V is a variable list. A list of distributive rational polynomials in r variables, where r=length(V), r ge 0, is read from the input stream. Any blanks preceding A are skipped. }

procedure DIRLWR (A,V,S: LIST);
{Distributive rational polynomial list write. V is a variable list. A list of distributive rational polynomials in r variables, where r=length(V), r ge 0, is written to the output stream. }

procedure DIRPAB (A: LIST): LIST;
{Distributive rational polynomial absolute value. A is a distributive rational polynomial. B is the absolute value of A.}

procedure DIRPDF (A,B: LIST): LIST;
{Distributive rational polynomial difference. A and B are distributive rational polynomials. C=A-B.}

procedure DIRPDM (A: LIST): LIST;
{Distributive rational polynomial derivation main variable. A is a distributive polynomial. B is the derivation of A with respect to its main variable.}

procedure DIRPDR (A,IL: LIST): LIST;
{Distributive rational polynomial derivation. A is a distributive polynomial. B is the derivation of A with respect to its i-th variable, 0 le i le DIPNOV(A).}

procedure DIRPEM (A,AL: LIST): LIST;
{Distributive rational polynomial evaluation of main variable. A is a distributive rational polynomial. a is a rational number. B(x1, ...,x(r-1))=A(x1, ...,x(r-1),a). }

procedure DIRPEV (A,IL,AL: LIST): LIST;
{Distributive rational polynomial evaluation of the i-th variable. A is a distributive rational polynomial, 1 le i le DIPNOV(A), a is a rational number. B(x1, ...,x(i-1),x(i+1), ...,xr)= A(x1, ...,x(i-1),a,x(i+1), ...,xr). }

procedure DIRPEX (A,NL: LIST): LIST;
{Distributive rational polynomial exponentiation. A is a distributive rational polynomial, n is a non-negative beta- integer. B=A**n. 0**0 is by definition a polynomial in zero variables. }

procedure DIRPHD (A,IL,NL: LIST): LIST;
{Distributive rational polynomial higher derivation. A is a distributive rational polynomial. B is the n-th derivation of A with respect to its i-th variable, 0 le i le DIPNOV(A). }

procedure DIRPLS (A: LIST): LIST;
{Distributive rational polynomial list sum. A is a circular list of distributive rational polynomials. B is the sum of all polynomials in A. }

procedure DIRPMC (A: LIST): LIST;
{Distributive rational polynomial monic. A and C are distributive rational polynomials, C=A/LBC(A) if A ne 0 C=0 if A eq 0. }

procedure DIRPMN (A: LIST): LIST;
{Distributive rational polynomial maximum norm. A is a distributive rational polynomial. b is the maximum norm of A.}

procedure DIRPNG (A: LIST): LIST;
{Distributive rational polynomial negative. B= -A.}

procedure DIRPON (A: LIST): LIST;
{Distributive rational polynomial one. A is a distributive rational polynomial. If A=1 then t=1, otherwise t=0.}

procedure DIRPPR (A,B: LIST): LIST;
{Distributive rational polynomial product. A and B are distributive rational polynomials. C=A*B.}

procedure DIRPQ (A,B: LIST): LIST;
{Distributive rational polynomial quotient. A and B are distributive rational polynomials. B is a non zero divisor of A. C=B/A.}

procedure DIRPQR (A,B: LIST; VAR Q,R: LIST);
{Distributive rational polynomial quotient and remainder. A and B are distributive rational polynomials with B ne 0. Q and R are unique distributive rational polynomials such that either B divides A, so Q=A/B and R=0 or B does not divide A, so A=B*Q+R with deg(R) lt deg(B). }

procedure DIRPRA (RL,KL,LL,EL: LIST): LIST;
{Distributive rational polynomial random. k, l and e are positive beta-digits. e is the maximal permitted exponent of A in any variable. A is a random distributive rational polynomial in r variables max norm of A lt 2**k and maximal l base coefficients. }

procedure DIRPRD (V: LIST): LIST;
{Distributive rational polynomial read. V is a variable list. A distributive rational polynomial A in r variables, where r=length(V), r ge 0, is read from the input stream. Any blanks preceding A are skipped. modified version, orginal version by G. E. Collins. }

procedure DIRPRP (A,BL: LIST): LIST;
{Distributive rational polynomial rational number product. Is a distributive rational polynomial, b is a rational number. C=A*b.}

procedure DIRPRQ (A,BL: LIST): LIST;
{Distributive rational polynomial rational number quotient. A is a distributive rational polynomial, b is a nonzero rational number. C=A/b.}

procedure DIRPSG (A: LIST): LIST;
{Distributive rational polynomial sign. A is a distributive rational polynomial. s is the sign of the leading base coefficient of A.}

procedure DIRPSM (A,B: LIST): LIST;
{Distributive rational polynomial sum. A and B are distributive rational polynomials. C=A+B. }

procedure DIRPSN (A: LIST): LIST;
{Distributive rational polynomial sum norm. A is a distributive rational polynomial. b is the sum norm of A.}

procedure DIRPSO (A: LIST): LIST;
{Distributive rational polynomial sort. A is a list of rational coefficients and exponent vectors, A is sorted into inverse lexicographical order, two terms with equal exponent vectors are added. }

procedure DIRPSU (A,IL,B: LIST): LIST;
{Distributive rational polynomial substitution. A and B are distributive rational polynomials, 1 le i le r=DIPNOV(A). E(x1, ...,x(i-1),x(i+1), ...,xr)=A(x1, ...,x(i-1), B(x1, ...,x(i-1),x(i+1), ...,xr),x(i+1), ...,xr). }

procedure DIRPSV (A,B: LIST): LIST;
{Distributive rational polynomial substitution for main variable. A and B are distributive rational polynomials. t=DIPNOV(A)-1. C(x1, ...,xt)=A(x1, ...,xt,B(x1, ...,xt)). }

procedure DIRPTM (A,HL: LIST): LIST;
{Distributive rational polynomial translation main variable. A is a distributive rational polynomial, h is a rational number. B(x1, ...xr)=A(x1, ...,x(r-1),x(r)+h). r=DIPNOV(A). }

procedure DIRPTR (A,HL,IL: LIST): LIST;
{Distributive rational polynomial translation. A is a distributive rational polynomial, h is a rational number, the i-th variable is translated. 1 le i le r=DIPNOV(A). B(x1, ...,xr)=A(x1, ...,x(i)+h, ...,xr).}

procedure DIRPWR (A,V,S: LIST);
{Distributive rational polynomial write. A is a distributive rational poynomial in r variables, r ge 0. V is a variable list for A. If S ge 0 then the coefficients are written by RNDWR else by RNWRIT. A is written in the output stream. Modified version, original version by G. E. Collins. }

procedure DIRPWV (A: LIST);
{Distributive rational polynomial write with standard variable list. A is a distributive rational poynomial. The standard variable list is used. A is written in the output stream.}

procedure DIRRAS (RL,KL,LL,EL,QL: LIST): LIST;
{Distributive rational polynomial, random sparse exponent vector. k, l and e are positive beta-digits. e is the maximal permitted exponent of A in any variable. A is a random distributive rational polynomial in r variables max norm of A lt 2**k and maximal l base coefficients. }

7.5 DIP Rational Number Polynomial

procedure RPABS (RL,A: LIST): LIST;
{Rational polynomial absolute value. A is a rational polynomial in r variables. B is the absolute value of A.}

procedure RPCONST (RL,A: LIST): LIST;
{Rational polynomial constant. A is a rational polynomial in r variables. If A is a non-zero rational number then t=1, otherwise t=0. }

procedure RPEXP (RL,A,NL: LIST): LIST;
{Rational polynomial exponentiation. A is a rational polynomial in r variables, r ge 0. n is a non-negative integer. B=A**n.}

procedure RPMON (RL,A: LIST): LIST;
{Rational polynomial monic. A is a rational polynomial in r variables. If A is non-zero then AP is the polynomial similar to A with LBCF(AP)=1. If A=0 then AP=0.}

procedure RPONE (RL,A: LIST): LIST;
{Rational polynomial one. A is a rational polynomial in r variables. If A=1 then t=1, otherwise t=0. }

procedure RPSIGN (RL,A: LIST): LIST;
{Rational polynomial sign. A is a rational polynomial in r variables. s is the sign of A.}

procedure RPLWRS (RL,A,V,S: LIST);
{Rational polynomial list write. A is a list of rational polynomial in r variables, r ge 0. V is a variable list for the polynomials in A. S is a decimal flag. A is written in the output stream in external canonical form.}

procedure RPWRTS (RL,A,V,S: LIST);
{Rational polynomial write. A is a rational polynomial in r variables, r ge 0. V is a variable list for A. S is a decimal flag. A is written in the output stream in external canonical form.}

procedure RUPEGC (A,B: LIST; VAR C,U,V: LIST);
{Rational univariate polynomial extended greatest common divisor. A and B are rational univariate polynomials. C=gcd(A,B). A*U+B*V=C, and, if deg(A/C) gt 0, then deg(V) lt deg(A/C), else deg(V)=0. Similarly, if deg(B/C) gt 0, then deg(U) lt deg(B/C), else deg(U)=0. If A=0, U=0. If B=0, V=0.}

procedure RUPGCD (A,B: LIST): LIST;
{Rational univariate polynomial greatest common divisor. A and B are rational univariate polynomials. C=gcd(A,B).}

procedure RUPHEG (A,B: LIST; VAR C,V: LIST);
{Rational univariate polynomial half-extended greatest common divisor. A and B are rational univariate polynomials. C=gcd(A,B). There exists a polynomial U such that A*U+B*V=C, and, if deg(A/C) gt 0, then deg(V) lt deg(A/C). If deg(A/C)=0, deg(V) is also 0. If B=0, V=0.}

procedure RUPLCM (A,B: LIST): LIST;
{Rational univariate polynomial least common multiple. A and B are rational univariate polynomials. C=LCM(A,B), a nonnegative rational univariate polynomial.}

7.6 DIP Termorder Optimization

procedure DIPDEM (A: LIST): LIST;
{Distributive polynomial degree matrix. A is a distributive polynomial. B is the degree matrix of A. }

procedure DIPDEV (A: LIST): LIST;
{Distributive polynomial degree vector. A is a distributive polynomial. N is the degree vector of A.}

procedure DIPLDM (A: LIST): LIST;
{Distributive polynomial list degree matrix. A is a list of distributive polynomials. B is the sum of all degree matrices of each element of A. }

procedure DIPTRM (A: LIST): LIST;
{Distributive polynomial terms. A is a distributive polynomial in r variables. T is a list of beta-integers each counting the terms in the respective variable.}

procedure DIPTYP (A: LIST): LIST;
{Distributive polynomial typ. A is a distributive polynomial in r variables. t is a rational number, t is the typ of A, 0 lt t le 1. }

procedure DIPVOP (P,V: LIST; VAR PP,VP: LIST);
{Distributive polynomial variable ordering optimisation. P and PP are lists of distributive polynomials. V and VP are variable lists. The optimal variable ordering for the polynomials in P is determined. The variables of the polynomials in P are permuted to produce PP. VP is the new variable list.}

procedure DIPVOPP (P,V: LIST; VAR PP,VP,PV: LIST);
{Distributive polynomial variable ordering optimization and permutation vector. P and PP are lists of distributive polynomials. V and VP are variable lists. The optimal variable ordering for the polynomials in P is determined. The variables of the polynomials in P are permuted to produce PP. VP is the new variable list, PV is the permutation to compute VP from V. }

procedure DMEVAD (A,E: LIST): LIST;
{Degree matrix exponent vector add. A is a degree matrix. E is an exponent vector. B=A + E. }

procedure HDIFDI (A: LIST; VAR B,FL: LIST);
{Homogeneous distributive polynomial from distributive polynomial. A is a distributive polynomial in r variables. s=r+1. If A is allready homogeneous then f=0 else f=1. B(xs,x1, ...,xr)=(xs)**(tdeg(A)) * A(x1/xs, ...,xr/xs). }

procedure INVPERM (perm: LIST):LIST;
{inverse permutation. perm is a permutation. The inverse permutation is returned, i.e. LPERM(LPERM(x,p),INVPERM(p))=x. }

procedure LBLXCO (U,V: LIST): LIST;
{List of beta integers lexicographical compare. U=(u1, ...,ur), V=(v1, ...vs) are lists of beta integers. t=0 if U eq V. t=1 if U gt V. t=-1 if U lt V. eq, gt, lt with respect to the lexicographical ordering of the beta integers. }

procedure LFCHECK (L, f: LIST): BOOLEAN;
{Linear form check. L is a linear form for term comparison, if L is empty EVORD is checked. f is a print flag, if f ¿0 then a message is written to the output stream. }

procedure PTERM (RL,A: LIST): LIST;
{Polynomial terms. A is a recursive polynomial in r variables. T is a list of beta-integers each counting the terms in the respective variable.}

procedure PTYP (RL,A: LIST): LIST;
{Polynomial typ. A is a recursive polynomial in r variables. t is a rational number, t is the PTYP of A, 0 lt t lt 1. }

procedure PVDEMA (A: LIST): LIST;
{Permutation vector for degree matrix. A is a degree matrix. P is a permutation vector. }

7.7 SAC Dense Polynomial

procedure DMPPRD (RL,ML,A,B: LIST): LIST;
{Dense modular polynomial product. A and B are polynomials in r variables over Z sub m, m a beta-integer, r ge 0. C=A*B.}

procedure DMPSUM (RL,ML,A,B: LIST): LIST;
{Dense modular polynomial sum. A and B are dense polynomials in r variables over Z sub m, m a beta-integer. C=A+B.}

procedure DMUPNR (PL,A,B: LIST): LIST;
{Dense modular univariate polynomial natural remainder. A and B are non-zero dense univariate polynomials over Z sub p, p a prime beta-integer, with deg(A) ge deg(B). C is the natural remainder of B. The list for A is modified.}

procedure DPFP (RL,A: LIST): LIST;
{Dense polynomial from polynomial. A is a polynomial in r variables, r ge 0. B is the result of converting A to dense polynomial representation.}

7.8 SAC Integer Polynomial System

procedure IPABS (RL,A: LIST): LIST;
{Integral polynomial absolute value. A is an integral polynomial in r variables. B is the absolute value of A.}

procedure IPCRA (M,ML,MLP,RL,A,AL: LIST): LIST;
{Integral polynomial chinese remainder algorithm. M is a positive integer. m is a positive beta-integer. gcd(M,m)=1. mp is the inverse of H sub m of M. A is an integral polynomial in r variables whose coefficients belong to Z prime sub M, r non-negative. a is a polynomial in r variables over Z sub m. AS is the unique integral polynomial in r variables with coefficients in Z prime sub MS, where MS=M*m, which is congruent to A modulo M and to a modulo m.}

procedure IPDER (RL,A,IL: LIST): LIST;
{Integral polynomial derivative. A is an integral polynomial in r variables. 1 le i le r. B is the derivative of A with respect to its i-th variable.}

procedure IPDIF (RL,A,B: LIST): LIST;
{Integral polynomial difference. A and B are integral polynomials in r variables, r ge 0. C=A-B.}

procedure IPDMV (RL,A: LIST): LIST;
{Integral polynomial derivative, main variable. A is an integral polynomial in r variables. B is the derivative of A with respect to its main variable.}

procedure IPEMV (RL,A,AL: LIST): LIST;
{Integral polynomial evaluation of main variable. A is an integral polynomial in r variables. a is an integer. B(x(1), ...,x(r-1))=A(x(1), ...,x(r-1),a).}

procedure IPEVAL (RL,A,IL,AL: LIST): LIST;
{Integral polynomial evaluation. A is an integral polynomial in r variables. 1 le i le r. a is an integer. B(x(1), ..., x(i-1),x(i+1), ...,x(r))=A(x(1), ...,x(i-1),a,x(i+1), ...,x(r)).}

procedure IPEXP (RL,A,NL: LIST): LIST;
{Integral polynomial exponentiation. A is an integral polynomial in r variables, r ge 0. n is a non-negative integer. B=A**n.}

procedure IPFCB (V: LIST): LIST;
{Integral polynomial factor coefficient bound. V is the degree vector of a non-zero integral polynomial A. b is a non-negative integer such that if b(1)* ...*b(k) divides A then the product of the infinity norms of the b(i) is less than or equal to 2**b times the infinity norm of A. Gelfonds bound is used.}

procedure IPFRP (RL,A: LIST): LIST;
{Integral polynomial from rational polynomial. A is a rational polynomial in r variables, r ge 0, each of whose base coefficients is an integer. B is a converted to integral polynomial representation.}

procedure IPGSUB (RL,A,SL,L: LIST): LIST;
{Integral polynomial general substitution. A is an integral polynomial in r variables, r ge 1. L is a list (b(1), ...,b(r)) of integral polynomials in s variables, s ge 1. C(y(1), ...,y(s))= A(b(1)(y(1), ...,y(s)), ...,b(r)(y(1), ...,y(s))).}

procedure IPHDMV (RL,A,KL: LIST): LIST;
{Integral polynomial higher derivative, main variable. A is an integral polynomial in r variables. k is a non-negative gamma-integer B is the k-th derivative of A with respect to its main variable.}

procedure IPIHOM (RL,D,A: LIST): LIST;
{Integral polynomial mod ideal homomorphism. D is a list (d sub 1, ..., d sub r-1) of non-negative beta-integers, r ge 0. A is an r-variate integral polynomial. B=A mod (x sub 1 ** d sub 1, ...,x sub r-1 ** d sub r-1).}

procedure IPIP (RL,AL,B: LIST): LIST;
{Integral polynomial integer product. a is an integer. B is an integral polynomial in r variables. C=a*B.}

procedure IPIPR (RL,D,A,B: LIST): LIST;
{Integral polynomial mod ideal product. D is a list (d sub 1, ..., d sub r-1) of non-negative beta-integers, r ge 1. A and B belong to Z(x sub 1, ...,x sub r-1,y)/(x sub 1 **d sub 1, ...,x sub r-1 ** d sub r-1). C=A*B.}

procedure IPIQ (RL,A,BL: LIST): LIST;
{Integral polynomial integer quotient. A is an integral polynomial in r variables. b is a non-zero integer which divides A. C=A/b.}

procedure IPMAXN (RL,A: LIST): LIST;
{Integral polynomial maximum norm. A is an integral polynomial in r variables. b is the maximum norm of A.}

procedure IPNEG (RL,A: LIST): LIST;
{Integral polynomial negative. A is an integral polynomial in r variables, r ge 0. B=-A.}

procedure IPONE (RL,A: LIST): LIST;
{Integral polynomial one. A is an integral polynomial in r variables. If A=1 then t=1, otherwise t=0.}

procedure IPPROD (RL,A,B: LIST): LIST;
{Integral polynomial product. A and B are integral polynomials in r variables, r ge 0. C=A*B.}

procedure IPPSR (RL,A,B: LIST): LIST;
{Integral polynomial pseudo-remainder. A and B are integral polynomials in r variables, B non-zero. C is the pseudo-remainder of A and B.}

procedure IPQ (RL,A,B: LIST): LIST;
{Integral polynomial quotient. A and B are integral polynomials in r variables, r ge 0. B is a non-zero divisor of A. C=A/B.}

procedure IPQR (RL,A,B: LIST; VAR Q,R: LIST);
{Integral polynomial quotient and remainder. A and B are integral polynomials in r variables with B non-zero. Q and R are the unique integral polynomials such that either B divides A, Q=A/B and R=0 or else B does not divide A and A=BQ+R with deg(R) minimal.}

procedure IPRAN (RL,KL,QL,N: LIST): LIST;
{Integral polynomial, random. k is a positive beta-digit. q is a rational number q1/q2 with 0 lt q1 le q2 lt beta. N is a list (n sub r, ...,n sub 1) of non-negative beta-digits, r ge 0. A is a random integral polynonial in r variables with deg sub i of a le n sub i for 1 le i le r. Max norm of A lt 2**k and q is the probability that any particular term of a has a non-zero coefficient.}

procedure IPREAD (VAR RL,A,V: LIST);
{Integral polynomial read. The integral polynomial A is read from the input stream. r, non-negative, is the number of variables of A and V is the variable list of A. Any number of preceding

blanks are skipped.}

procedure IPSIGN (RL,A: LIST): LIST;
{Integral polynomial sign. A is an integral polynomial in r variables. s is the sign of A.}

procedure IPSMV (RL,A,B: LIST): LIST;
{Integral polynomial substitution for main variable. A is an integral polynomial in r variables, x(1), ...,x(r). B is an integral polynomial in x(1), ...,x(r-1). C(x(1), ...,x(r-1))= A(x(1), ...,x(r-1),B(x(1), ...,x(r-1))).}

procedure IPSUB (RL,A,IL,B: LIST): LIST;
{Integral polynomial substitution. A is an integral polynomial in r variables, x(1), ...,x(r). 1 le i le r. B is an integral polynomial in x(1), ...,x(i-1). C(x(1), ...,x(i-1),x(i+1), ..., x(r))=A(x(1), ...,x(i-1),B(x(1), ...,x(i-1)),x(i+1), ..., x(r)).}

procedure IPSUM (RL,A,B: LIST): LIST;
{Integral polynomial sum. A and B are integral polynomials in r variables, r ge 0. C=A+B.}

procedure IPSUMN (RL,A: LIST): LIST;
{Integral polynomial sum norm. A is an integral polynomial in r variables, r non-negative. b is the sum norm of A.}

procedure IPTPR (RL,D,A,B: LIST): LIST;
{Integral polynomial truncated product. D is a list (d sub 1, ...,d sub r) of non-negative beta-integers, r ge 1. A and B belong to Z(x sub 1, ...,x sub r)/(x sub 1 **d sub 1, ...,x sub r ** d sub r). C=A*B.}

procedure IPTRAN (RL,A,T: LIST): LIST;
{Integral polynomial translation. A is an integral polynomial in r variables, r ge 1. T is a list (tr, ...,t1) of integers. B(x1, ...,xr)=A(x1+t1, ...,xr+tr).}

procedure IPTRMV (RL,A,HL: LIST): LIST;
{Integral polynomial translation, main variable. A is an integral polynomial in r variables, r ge 1. h is an integer. B(x1, ...,xr)=A(x1, ...,x(r-1),xr+h).}

procedure IPTRUN (RL,D,A: LIST): LIST;
{Integral polynomial truncation. D is a list (d sub 1, ...,d sub r) of non-negative beta-integers, r ge 0. A is an r-variate integral polynomial. B=A mod (x sub 1 ** d sub 1, ..., x sub r ** d sub r).}

procedure IPWRIT (RL,A,V: LIST);
{Integral polynomial write. A is an integral polynomial in r variables, r ge 0. V is a variable list for A. A is written in the output stream in external canonical form.}

procedure IUPBEI (A,CL,ML: LIST): LIST;
{Integral univariate polynomial binary rational evaluation, integer output. A is a univariate integral polynomial. c is an integer. m is a non-negative beta-integer. b=2**(n*m)*A(c/2**m) where n=deg(A). b is an integer.}

procedure IUPBES (A,AL: LIST): LIST;
{Integral univariate polynomial binary rational evaluation of sign. A is a univariate polynomial. a is a binary rational number. s=sign(A(a)).}

procedure IUPBHT (A,KL: LIST): LIST;
{Integral univariate polynomial binary homothetic transformation. A is a non-zero univariate integral polynomial. k is a gamma-integer. B(x)=2**(-h)*A(2**k*x) where h is uniquely determined so that B is an integral polynomial not divisible by 2.}

procedure IUPBRE (A,AL: LIST): LIST;
{Integral univariate polynomial binary rational evaluation. A is a univariate integral polynomial. a is a binary rational number. B=A(a), a binary rational number.}

procedure IUPCHT (A: LIST): LIST;
{Integral univariate polynomial circle to half-plane transformation. A is a non-zero univariate integral polynomial. Let n=deg(A). Then B(x)=(x+1)**n*A(1/(x+1)), a univariate integral polynomial.}

procedure IUPNT (A: LIST): LIST;
{Integral univariate polynomial negative transformation. A is a univariate integral polynomial. B(x)=A(-x).}

procedure IUPTPR (NL,A,B: LIST): LIST;
{Integral univariate polynomial truncated product. n is a non- negative integer. A and B are integral univariate polynomials. C(x)=A(x)*B(x) (modulo x**n) and C=0 or deg(C) lt n.}

procedure IUPTR (A,HL: LIST): LIST;
{Integral univariate polynomial translation. A is a univariate integral polynomial. h is an integer. B(x)=A(x+h).}

procedure IUPTR1 (A: LIST): LIST;
{Integral univariate polynomial translation by 1. A is a univariate integral polynomial. B(x)=A(x+1).}

7.9 SAC Modular Polynomial

procedure MIPDIF (RL,M,A,B: LIST): LIST;
{Modular integral polynomial difference. M is a positive integer. A and B are polynomials in r variables over Z sub M, r ge 0. C=A-B.}

procedure MIPFSM (RL,M,A: LIST): LIST;
{Modular integral polynomial from symmetric modular. M is a positive integer. A is a polynomial in r variables over Z prime sub M, r ge 0. B belongs to Z sub M (x, ...,x sub r) with B=A (modulo M).}

procedure MIPHOM (RL,M,A: LIST): LIST;
{Modular integral polynomial homomorphism. A is an integral polynomial in r variables, r ge 0. M is a positive integer. B=H sub M (A), a polynomial in r variables over Z sub M.}

procedure MIPIPR (RL,M,D,A,B: LIST): LIST;
{Modular integral polynomial mod ideal product. D is a list (d sub 1, ...,d sub r-1) of non-negative beta-integers, r ge 1. M is a positive integer. A and B belong to Z sub M (x sub 1, ...,x sub r-1,y)/(x sub 1 ** d sub 1, ...,x sub r-1 ** d sub r-1). C=A*B.}

procedure MIPNEG (RL,M,A: LIST): LIST;
{Modular integral polynomial negation. M is a positive integer. A is a polynomial in r variables over Z sub M, r ge 0. B=-A.}

procedure MIPPR (RL,M,A,B: LIST): LIST;
{Modular integral polynomial product. M is a positive integer. A and B are polynomials in r variables over Z sub M, r ge 0. C=A*B.}

procedure MIPRAN (RL,M,QL,N: LIST): LIST;
{Modular integral polynomial, random. M is a positive integer. q is a rational number q1/q2 with 0 lt q1 le q2 lt beta. N is a list (n sub r, ...,n sub 1) of non-negative beta-digits, r ge 0. A is a random polynomial in r variables over Z sub M with deg sub i of A le n sub i for 1 le i le r. q is the probability that any particular term of A has a non-zero coefficient.}

procedure MIPSUM (RL,M,A,B: LIST): LIST;
{Modular integral polynomial sum. M is a positive integer. A and B are polynomials in r variables over Z sub M, r ge 0. C=A+B.}

procedure MIUPQR (M,A,B: LIST; VAR Q,R: LIST);
{Modular integral univariate polynomial quotient and remainder. M is a positive integer. A and B belong to Z sub M (x) with LDCF(B) a unit. Q and R are the unique elements of Z sub M (x) such that A=B*Q+R with either R=0 or DEG(R) lt DEG(B).}

procedure MMPIQR (RL,M,D,A,B: LIST; VAR Q,R: LIST);
{Modular monic polynomial mod ideal quotient and remainder. M is a positive integer. D is a list (d sub 1, ...,d sub r-1) of non-nega- tive beta-integers, r ge 1. A and B belong to Z sub M(x sub 1, ...,x sub r-1,y)/(x sub 1 ** d sub 1, ...,x sub r-1 ** d sub r-1), with B monic. A=B*Q+R, deg sub y of R lt deg sub y of B unless B divides A, in which case R=0, with Q,R belonging to Z sub M (x sub 1, ...,x sub r-1,y)/(x sub 1 ** d sub 1, ...,x sub r-1 ** d sub r-1).}

procedure MPDIF (RL,ML,A,B: LIST): LIST;
{Modular polynomial difference. A and B are polynomials in r variables over Z sub m, m a beta-integer. C=A-B.}

procedure MPEMV (RL,ML,A,AL: LIST): LIST;
{Modular polynomial evaluation of main variable. A is a polynomial in r variables over Z sub m, m a beta-integer. a is an element of Z sub m. B(x(1), ...,x(r-1))=A(x(1), ...,x(r-1),a).}

procedure MPEVAL (RL,ML,A,IL,AL: LIST): LIST;
{Modular polynomial evaluation. A is a polynomial in r variables over Z sub m, m a beta-integer. 1 le i le r. a is an element of Z sub m. B(x(1), ...,x(i-1),x(i+1), ...,x(r))= A(x(1), ...,x(i-1),a,x(i+1), ...,x(r)).}

procedure MPEXP (RL,ML,A,NL: LIST): LIST;
{Modular polynomial exponentiation. A is a polynomial in r variables over Z sub m, m a beta-integer. n is a non-negative integer. B=A**n.}

procedure MPHOM (RL,ML,A: LIST): LIST;
{Modular polynomial homomorphism. A is an integral polynomial in r variables, r ge 0. m is a positive beta-integer. B is the image of A under the homomorphism H sub m, a polynomial in r variables over Z sub m.}

procedure MPINT (PL,B,BL,BLP,RL,A,A1: LIST): LIST;
{Modular polynomial interpolation. p is a prime beta-integer. B is a univariate polynomial over Z sub p. b is an element of Z sub p such that B(b) ne 0 and bp=B(b)**-1. A is a polynomial over Z sub p in r variables, r ge 1, with A=0 or the degree of A in x(1) less than the degree of B. A1 is a polynomial over Z sub p in r-1 variables. AS(x(1), ...,x(r)) is the unique polynomial over Z sub p such that AS(x(1), ...,x(r)) is congruent to A(x(1), ...,x(r)) modulo B(x(1)), AS(b,x(2), ...,x(r))=A1(x(2), ...,x(r)) and the degree of AS in x(1) is less than or equal to the degree of B.}

procedure MPMDP (RL,PL,AL,B: LIST): LIST;
{Modular polynomial modular digit product. a is an element of Z sub p, p a prime beta-integer. B is a polynomial in r variables over Z sub p. C=a*B.}

procedure MPMON (RL,PL,A: LIST): LIST;
{Modular polynomial monic. A is a polynomial in r variables over Z sub p, p a prime beta-integer. If A is non-zero then AP is the polynomial similar to A with LBCF(AP)=1. If A=0 then AP=0.}

procedure MPNEG (RL,ML,A: LIST): LIST;
{Modular polynomial negative. A is a polynomial in r variables over Z sub m, m a beta-integer. B=-A.}

procedure MPPROD (RL,ML,A,B: LIST): LIST;
{Modular polynomial product. A and B are polynomials in r variables over Z sub m, m a beta-integer, r ge 0. C=A*B.}

procedure MPPSR (RL,PL,A,B: LIST): LIST;
{Modular polynomial pseudo-remainder. A and B are polynomials in r variables over Z sub p, p a prime beta-integer, with B non-zero. C is the pseudo-remainder of A and B.}

procedure MPQ (RL,PL,A,B: LIST): LIST;
{Modular polynomial quotient. A and B are polynomials in r variables over Z sub p, p a prime beta-integer, r ge 0. B is a non-zero divisor of A. C=A/B.}

procedure MPQR (RL,PL,A,B: LIST; VAR Q,R: LIST);
{Modular polynomial quotient and remainder. A and B are polynomials un r variables over Z sub p, p a prime beta-integer, with B non-zero. Q and R are the unique polynomials such that either B divides A, Q=A/B and R=0 or else B does not divide A and A=BQ+R with DEG(R) minimal.}

procedure MPRAN (RL,ML,QL,N: LIST): LIST;
{Modular polynomial, random. m is a positive beta-integer. q is a rational number q1/q2 with 0 lt q1 le q2 lt beta. N is a list (n sub r, ...,n sub 1) of non-negative beta-digits, r ge 0. A is a random polynomial in r variables over Z sub m with deg sub i of A le n sub i for 1 le i le r. q is the probability that any particular term of A has a non-zero coefficient.}

procedure MPSUM (RL,ML,A,B: LIST): LIST;
{Modular polynomial sum. A and B are polynomials in r variables over Z sub m, m a beta-integer. C=A+B.}

procedure MPUP (RL,ML,CL,A: LIST): LIST;
{Modular polynomial univariate product. A is a polynomial in r variables, r ge 1, over Z sub m, m a positive beta-integer. c is a univariate polynomial over Z sub m. B(x(1), ...,x(r)) = c(x(1))*A(x(1), ...,x(r)).}

procedure MPUQ (RL,PL,A,BL: LIST): LIST;
{Modular polynomial univariate quotient. A is a polynomial in r variables, r ge 2, over Z sub p, p a prime beta-integer. b is a non-zero univariate polynomial over Z sub p which divides A. C(x(1), ...,x(r))=A(x(1), ...,x(r))/b(x(1)).}

procedure MUPDER (ML,A: LIST): LIST;
{Modular univariate polynomial derivative. m is a beta-integer. A is a univariate polynomial over Z sub m. B is the derivative of A, a univariate polynomial over Z sub m.}

procedure MUPRAN (PL,NL: LIST): LIST;
{Modular univariate polynomial, random. A is a random univariate polynomial of degree n over Z(p).}

procedure SMFMIP (RL,M,A: LIST): LIST;
{Symmetric modular from modular integral polynomial. M is a positive integer. A is a polynomial in r variables over Z sub M, r ge 0. B belongs to Z prime sub M (x1, ...,x sub r) with B=A (modulo M).}

procedure VMPIP (RL,ML,A,B: LIST): LIST;
{Vector of modular polynomial inner product. A and B are vectors of modular polynomials in r variables over Z sub m, r non-negative, m a beta-integer. C is the inner product of A and B.}

7.10 SAC Polynomial System

procedure PBIN (AL1,EL1,AL2,EL2: LIST): LIST;
{Polynomial binomial. a1 and a2 are elements of a coefficient ring R. e1 and e2 are non-negative beta-integers e1 gt e2. A is the polynomial A(x)=a1*x**e1+a2*x**e2, a univariate polynomial over R.}

procedure PCL (A: LIST): LIST;
{Polynomial coefficient list. A is a non-zero polynomial. L is the list (a(n),a(n-1), ...,a(0)) where n=DEG(A) and A(x)=a(n)*x**n+ a(n-1)*x**(n-1)+ ...+a(0).}

procedure PDBORD (A: LIST): LIST;
{Polynomial divided by order. A is a non-zero polynomial. B(x)= A(x)/x**k where k is the order of A.}

procedure PDEG (A: LIST): LIST;
{Polynomial degree. A is a polynomial. n is the degree of A.}

procedure PDEGSV (RL,A,IL: LIST): LIST;
{Polynomial degree, specified variable. A is a polynomial in r variables, r ge 1. 1 le i le r. n is the degree of A in the i-th variable.}

procedure PDEGV (RL,A: LIST): LIST;
{Polynomial degree vector. A is a polynomial A(x(1), ...,x(r)) in r variables. V is the list (v(r), ...,v(1)) where v(i) is the degree of a in x(i).}

procedure PDPV (RL,A,IL,NL: LIST): LIST;
{Polynomial division by power of variable. A is a polynomial in r variables. 1 le i le r and n is a beta-integer such that x sub i sup n divides A. B eq A/x sub i sup n.}

procedure PFDP (RL,A: LIST): LIST;
{Polynomial from dense polynomial. A is a dense polynomial in r variables, r ge 0. B is the result of converting A to recursive polynomial representation.}

procedure PINV (RL,A,KL: LIST): LIST;
{Polynomial introduction of new variables. A is a polynomial in r variables, r ge 0. k ge 0. B(y(1), ...,y(k),x(1), ...,x(r)) =A(x(1), ...,x(r)).}

procedure PLBCF (RL,A: LIST): LIST;
{Polynomial leading base coefficient. A is a polynomial in r variables. a is the leading base coefficient of A.}

procedure PLDCF (A: LIST): LIST;
{Polynomial leading coefficient. A is a polynomial. a is the leading coefficient of A.}

procedure PMDEG (A: LIST): LIST;
{Polynomial modified degree. A is a polynomial. If A=0 then n=-1 and otherwise n=DEG(A).}

procedure PMON (AL,EL: LIST): LIST;
{Polynomial monomial. a is an element of a coefficient ring R. e is a non-negative beta-integer. A is the polynomial A(x)=a*x**e, a univariate polynomial over R.}

procedure PMPMV (A,KL: LIST): LIST;
{Polynomial multiplication by power of main variable. A is a polynomial in r variables, r ge 1. k is a non-negative integer. B(x sub 1 , ..., x sub r) eq A(x sub 1 , ..., x sub r) * x sub r sup k .}

procedure PORD (A: LIST): LIST;
{Polynomial order. A is a non-zero polynomial. k is the order of A. that is, if A(x)=a(n)*x**n+ ...+a(0), then k is the smallest integer such that a(k) ne 0.}

procedure PRED (A: LIST): LIST;
{Polynomial reductum. A is a polynomial. B is the reductum of A.}

procedure PRT (A: LIST): LIST;
{Polynomial reciprocal transformation. A is a non-zero polynomial. let n=DEG(A). Then B(x)=x**n*A(1/x), where x is the main variable of A.}

procedure PTBCF (RL,A: LIST): LIST;
{Polynomial trailing base coefficient. A is an r-variate polynomial, r ge 0. a=trailing base coefficient of A.}

procedure PUFP (RL,A: LIST): LIST;
{Polynomial, univariate, from polynomial. A is an r-variate polynomial, r ge 0. B, a univariate polynomial, equals A(0, ...,0,x).}

procedure VCOMP (U,V: LIST): LIST;
{Vector comparison. U=(u(1), ...,u(r)) and V=(v(1), ...,v(r)) are lists of beta-integers with common length r ge 1. If U=V then t=0. If U is not equal to V then t=1 if u(i) le v(i) for all i and t=2 if v(i) le u(i) for all i. Otherwise t=3.}

procedure VLREAD (): LIST;
{Variable list read. V, a list of variables, is read from the input stream. Any preceding blanks are skipped.}

procedure VLSRCH (VL,V: LIST): LIST;
{Variable list search. v is a variable. V is a list of variables (v(1), ...,v(n)), n non-negative. If v=v(j) for some j then i=j. Otherwise i=0.}

procedure VLWRIT (V: LIST);
{Variable list write. V, a list of variables, is written in the output stream.}

procedure VMAX (U,V: LIST): LIST;
{Vector maximum. U=(u(1), ...,u(r)) and V=(v(1), ...,v(r)) are lists of beta-integers with common length r ge 1. W=(w(1), ..., w(r)) where w(i)=MAX(u(i),v(i)).}

procedure VMIN (U,V: LIST): LIST;
{Vector maximum. U=(u(1), ...,u(r)) and V=(v(1), ...,v(r)) are lists of beta-integers with common length r ge 1. W=(w(1), ..., w(r)) where w(i)=MIN(u(i),v(i)).}

procedure VREAD (): LIST;
{Variable read. The variable v is read from the input stream. Any number of preceding blanks are skipped.}

7.11 SAC Rational Polynomial

procedure RPDIF (RL,A,B: LIST): LIST;
{Rational polynomial difference. A and B are rational polynomials in r variables, r ge 0. C=A-B.}

procedure RPEMV (RL,A,BL: LIST): LIST;
{Rational polynomial evaluation, main variable. A is a rational polynomial in r variables, r gt 0. b is a rational number. C(x(1), ...,x(r-1))=A(x(1), ...,x(r-1),b).}

procedure RPFIP (RL,A: LIST): LIST;
{Rational polynomial from integral polynomial. A is an integral polynomial in r variables, r ge 0.}

procedure RPIMV (RL,A: LIST): LIST;
{Rational polynomial integration, main variable. A is a rational polynomial in r variables, r gt 0. B is the integral of A with respect to its main variable. The constant of integration is 0.}

procedure RPNEG (RL,A: LIST): LIST;
{Rational polynomial negative. A is an rational polynomial in r variables, r ge 0. B=-A.}

procedure RPPROD (RL,A,B: LIST): LIST;
{Rational polynomial product. A and B are rational polynomials in r variables, r ge 0. C=A*B.}

procedure RPQR (RL,A,B: LIST; VAR Q,R: LIST);
{Rational polynomial quotient and remainder. A and B are rational polynomials in r variables with B non-zero. Q and R are the unique rational polynomials such that either B divides A, Q=A/B and R=0 or else B does not divide A and A=BQ+R with DEG(R) minimal.}

procedure RPREAD (VAR RL,A,V: LIST);
{Rational polynomial read. The rational polynomial A is read from the input stream. r ge 0 is the number of variables of A and V is the variable list of A. Any number of preceding blanks are skipped.}

procedure RPRNP (RL,AL,B: LIST): LIST;
{Rational polynomial rational number product. B is a rational polynomial in r variables, r ge 0. a is a rational number. C=a*B.}

procedure RPSUM (RL,A,B: LIST): LIST;
{Rational polynomial sum. A and B are rational polynomials in r variables, r ge 0. C=A+B.}

procedure RPWRIT (RL,A,V: LIST);
{Rational polynomial write. A is a rational polynomial in r variables, r ge 0. V is a variable list for A. A is written in the output stream in external canonical form.}

Chapter 8

Module Algorithms

8.1 G-Symmetric Integral Polynomial System

procedure GSYINF ();
{G-Symmetric Polynomial System Information. }

procedure GSYPGR (M: GAMMAINT): LIST;
{G-Symmetric Permutation Group Read. Input of producing elements for a permutation group with M variables. }

procedure GSYPGW (PG: LIST);
{G-Symmetric Permutation Group Write. Output of producing elements for the permutation group PG. }

procedure GINORP (PG, MO: LIST): LIST;
{G-Symmetric Integral Orbit Polynomial. The PG-symmetric orbit polynomial of the monomial MO is computed with respect to the permutation group PG. }

procedure GSYORD (PG: LIST): GAMMAINT;
{G-Symmetric Permutation Group Order. The order of the permutation group PG is computed. }

procedure GSYNSP (PG: LIST);
{G-Symmetric Number of Special Polynomials. The number of special polynomials is computed. }

procedure GSYSPG (N: GAMMAINT): LIST;
{Symmetric Permutation Group. The symmetric permutation group is computed for N variables. }

procedure GINOPL (PG, ML: LIST): LIST;
{G-Symmetric Integral Orbit Polynomial List. The PG-symmetric orbit polynomial list is calculated from the monomial list ML and the permutation group PG. }

procedure GINCUT (PG, POL: LIST; VAR POL_1, POL_2: LIST);
{G-Symmetric Integral Polynomial Cut. The Polynomial POL is splitted up into the G-symmetric polynomial POL_1 and the remainder polynomial POL_2 with respect to the permutation group PG and the termorder. }

procedure GINCHK (PG, BASE, POL: LIST): LIST;
{G-Symmetric Integral Polynomial Check. The original polynomial is computed from the polynomial POL and the PG-symmetric base polynomials BASE with respect to the permutation group PG. }

procedure GINCHKBAS (VAR BASE, POL: LIST);
{G-Symmetric Integral Base Check. The procedure removes not used base orbit polynomials from BASE and make an update of POL. }

procedure GSYTWG (TERM1, TERM2: LIST): GAMMAINT;
{G-Symmetric Term Weight. The weigth between TERM1 and TERM2 is computed w.r.t. the default term order. The result is 1, if wg(TERM1) ¿ wg(TERM2), and 0, if wg(TERM1) = wg(TERM2); otherwise the result is -1. }

procedure GSYMLT (N: GAMMAINT): GAMMAINT;
{G-Symmetric Multilinear Terms. The result of the preocedure ist a list off all multilinear terms in N variables. }

procedure GSYADD (TERM: LIST): LIST;
{G-Symmetric Term Adder. The result of the procedure is the next highest descend term after TERM. }

procedure GINRED (PG, POL: LIST; VAR BASE, BASE_POL, REM_POL: LIST);
{G-Symmetric Integral Polynomial Reduction. The PG-symmetric polynomial BASE_POL, which is the PG-symmetric polynomial reconstruction with respect to the base polynomials BASE, and the remainder polynomial REM_POL are computed from the polynomial POL with respect to the permutation group PG. }

procedure GINBAS (PG: LIST): LIST;
{G-Symmetric Integral Base Construction. The PG-symmetric base polynomials for the permutation group PG are computed. }

8.2 G-Symmetric Rational Polynomial System

procedure GRNORP (PG, MO: LIST): LIST;
{G-Symmetric Rational Orbit Polynomial. The PG-symmetric orbit polynomial of the monomial MO is computed with respect to the permutation group PG. }

procedure GRNOPL (PG, ML: LIST): LIST;
{G-Symmetric Rational Orbit Polynomial List. The PG-symmetric orbit polynomial list is calculated from the the monomial list ML and the permutation group PG. }

procedure GRNCUT (PG, POL: LIST; VAR POL_1, POL_2: LIST);
{G-Symmetric Rational Polynomial Cut. The Polynomial POL is splitted up into the G-symmetric polynomial POL_1 and the remainder polynomial POL_2 with respect to the permutation group PG and the termorder. }

procedure GRNCHK (PG, BASE, POL:LIST): LIST;
{G-Symmetric Rational Polynomial Check. The original polynomial is computed from the polynomial POL and the PG-symmetric base polynomials BASE with respect to the permutation group PG. }

procedure GRNCHKBAS (VAR BASE, POL: LIST);
{G-Symmetric Rational Base Check. The procedure removes not used base orbit polynomials from BASE and make an update of POL. }

procedure GRNRED (PG, POL: LIST; VAR BASE, BASE_POL, REM_POL: LIST);
{G-Symmetric Rational Polynomial Reduction. The PG-symmetric polynomial BASE_POL, which is the PG-symmetric polynomial reconstruction with respect to the base polynomials BASE, and the remainder polynomial REM_POL are computed from the polynomial POL with respect to the permutation group PG. }

procedure GRNBAS (PG: LIST): LIST;
{G-Symmetric Rational Base Construction. The PG-symmetric base polynomials for the permu-

tation group PG are computed. }

procedure GRNGGB (PG: LIST): LIST;
{G-Symmetric Rational Base Construction (Buchberger-Algorithm). The PG-symmetric base polynomials for the permutation group PG are computed with the Buchberger-Algorithm. }

8.3 GSYM Input

procedure GSDREAD (): LIST;
{G-symmetric descriptor read. — to do —: remove }

procedure GSPREAD (): LIST;
{G-symmetric polynomial read. — to do —: remove }

procedure GSRDREAD (): LIST;
{G-symmetric rational descriptor read. — to do —: remove }

procedure GSRREAD (): LIST;
{G-symmetric rational polynomial read. — to do —: remove }

8.4 MAS Linear Algebra Integer

procedure IUM (m, n : LIST): LIST;
{Integer unit matrix. m, n integer. An (m x n) integer unit matrix is returned. }

procedure IVWRITE (A : LIST);
{Integer vector write. A is an integer vector. A is written to the output stream. }

procedure IMWRITE (A : LIST);
{Integer matrix write. A is an integer matrix. A is written to the output stream. }

procedure IVVDIF (A, B : LIST): LIST;
{Integer vector difference. A and B are integer vectors. The integer vector C = A - B is returned. }

procedure IKM (A, B : LIST): LIST;
{Integer vector component product. IKM returns the difference of the product of the integer vector A with FIRST(B) and the product of the integer vector B with FIRST(A). C = A * FIRST(B) - B * FIRST(A). C is an integer vector. }

procedure IVVSUM (A, B : LIST): LIST;
{Integer vector vector sum. A and B are integer vectors. An integer vector C = A + B is returned. }

procedure IVSVSUM (A, B : LIST): LIST;
{Integer vector scalar and vector sum. A and B are integer vectors. An integer vector C = A + FIRST(B) is returned. }

procedure IVSSUM (A : LIST): LIST;
{Integer vector scalar sum. A is an integer vector. An integer C = a1 + a2 + ... + an is returned. }

procedure IVSVPROD (A, B : LIST): LIST;
{Integer vector scalar and vector product. A and B are integer vectors. An integer vector C = (a1*FIRST(B), ..., an*FIRST(B)) is returned. }

procedure IVVPROD (A, B : LIST): LIST;
{Integer vector vectors product. A and B are integer vectors. An integer vector C = (a1*b1, ..., an*bn) is returned. }

procedure IVSPROD (A, B : LIST): LIST;
{Integer vector scalar product. A and B are integer vectors. An integer C = a1*b1 + ... + an*bn is returned. }

procedure IVMAX (M : LIST): LIST;
{Integer vector maximum norm. M is an integer vector. An integer a = maximum absolute value M(i) is returned. }

procedure IVLC (a, A, b, B : LIST): LIST;
{Integer vector linear combination. A and B are integer vectors. a and b are integers. An integer vector C = a*A + b*B is returned. }

procedure IVSQ (a, A: LIST): LIST;
{Integer vector scalar quotient. A is an integer vector. a is an integer. An integer vector C = A/a is returned. a must divide each element of A exactly. }

procedure IVFRNV (A: LIST): LIST;
{Integer vector from rational number vector. A is a rational number vector. A is muliplied by a common multiple of its denominators, then the denominators are removed. An integer vector C = lcm(denom(A)) * A is returned. }

procedure IVFRNV1 (A, B : LIST; VAR C, D: LIST);
{Integer vector from rational number vector. A and B are rational number vectors. A and B are muliplied by a common multiple of their denominators, then the denominators are removed. C and D are integer vectors, such that C = lcm(denom(A),denom(B))*A and D = lcm(denom(A),denom(B))*B. }

procedure IMPROD (A, B : LIST): LIST;
{Integer matrix product. A and B are integer matrices. An integer matrix C = A * B is returned, if the number of coloums of A is equal to the number of rows of B, otherwise the empty matrix is returned. }

procedure IMSUM (A, B : LIST): LIST;
{Integer matrix sum. A and B are integer matrices. An integer matrix C = A + B is returned. }

procedure IMDIF (A, B : LIST): LIST;
{Integer matrix difference. A and B are integer matrices. An integer matrix C = A - B is returned. }

procedure ISMPROD (A, B : LIST): LIST;
{Integer scalar and matrix product. A is an integer matrix. B is an integer. An integer matrix C = A * B is returned. }

procedure IMMAX (M : LIST): LIST;
{Integer matrix maximum norm. M is an integer matrix. An integer a = maximum absolute value M(i,j) is returned. }

procedure IMFRNM (A : LIST): LIST;
{Integer matrix from rational number matrix. A is a rational number row matix. The rows of A are muliplied by a common multiple of its denominators, then the denominators are removed. An integer matix C is returned, such that for all rows C(i) = lcm(denom(A(i))) * A(i). }

procedure IMFRNM1 (A, B : LIST; VAR C, D: LIST);
{Integer matrix from rational number matrix. A is a rational number row matix. B is a rational number column matix. The rows of A and the rows of B are muliplied by a common multiple of their denominators, then the denominators are removed. C and D are integer matices, such that C(i) = lcm(denom(A(i)),B(i)) * A(i) and D(i) = lcm(denom(A(i)),B(i)) * B(i). }

procedure IMGELUD (M : LIST; VAR L, U: LIST);
{Integer matrix Gaussian elimination LU-decomposition. M is an integer matrix represented rowwise. L is a lower triangular integer matrix represented columnwise. U is an upper triangular integer matrix represented rowwise. M = L * U for appropriate modifications of L and U. The

pivot operations and exact division factors are also recorded in L. }

procedure IMLT (L, b : LIST): LIST;
{Integer lower triangular matrix transformation. L is a lower triangular integer matrix represented columnwise as generated by IMGELUD. b is an integer vector. An integer vector u = L * b is returned, such that if M * x = b and M = L * U, then U * x = u. }

procedure IMUT (U, b : LIST): LIST;
{Integer upper triangular matrix transformation. U is an upper triangular integer matrix represented rowwise as generated by IMGELUD. b is an integer vector b = L * b' as generated by IMLT. A rational number (!) vector x, such that U * x = b is returned. If no such x exists, then an empty vector is returned. If more than one such x exists, then for free x(i), x(i) = 0 is taken. }

procedure IMGE (M : LIST): LIST;
{Integer matrix Gaussian elimination. M is a (n x m) integer matrix. A (n x m) integer matrix resulting from Gaussian elimination is returned. IMGELUD is called. }

procedure IMSDS (L, U, b : LIST): LIST;
{Integer matrix solve decomposed system. L is a lower triangular integer matrix represented columnwise, U is an upper triangular integer matrix represented rowwise. L and U as generated by IMGELUD. If M = L * U, then a rational number (!) vector x, such that M * x = b is returned. If no such x exists, then an empty vector is returned. If more than one such x exists, then for free x(i), x(i) = 0 is taken. }

procedure RNMINVI (A : LIST): LIST;
{Rational number matrix inversion, integer algorithm. A is a rational number matrix represented rowwise. If it exists, the inverse matrix of A is returned, otherwise an empty matrix is returned. The integer Gaussian elimination IMGELUD is used. }

procedure IMUNS (U : LIST): LIST;
{Integer matrix upper triangular matrix solution null space. U is an upper triangular integer matrix represented rowwise as generated by IMGELUD. A matrix X of linear independent rational number vectors x is returned, such that for each x in X, U * x = 0 holds. If only x = 0 satisfies the condition U * x = 0, then the matrix X is empty. }

procedure IMDETL (M : LIST): LIST;
{Integer matrix determinant, using Laplace expansion. M is an integer matrix. The determinant of M is returned. }

procedure IMDET (M : LIST): LIST;
{Integer matrix determinant, using Gaussian elimination. M is an integer matrix. The determinant of M is returned. }

8.5 MAS Linear Algebra Rational Number

procedure MDIM (M : LIST): LIST;
{Matrix dimension. M is a matrix. MDIM returns max(row, column) of M. }

procedure MGET (M, k, l : LIST): LIST;
{Matrix get. M is a matrix. k, l are integers, 0 le k le rows(M), 0 le l le columns(M). MGET returns the element M(k,l) of matrix M. }

procedure MSET (M, k, l, x : LIST): LIST;
{Matrix set. M is a matrix. k, l are integers, 0 le k le rows(M), 0 le l le columns(M). MSET sets the element M(k,l) to x. The new matrix is returned. }

procedure VDELEL (V, i : LIST): LIST;
{Vector delete element. V is a vector. The i-th element of V is deleted. 0 le i le length(V). }

procedure MDELCOL (M, i : LIST): LIST;
{Matrix delete column. M is a vector of row vectors. In each row the i-th element is deleted, 0 le i le columns(M). The new matrix is returned. }

procedure MMINOR (M, i, j : LIST): LIST;
{Matrix minor. M is a vector of row vectors. The i-th column, 0 le i le rows(M), and in each remaining row the j-th element, 0 le j le columns(M), is deleted. }

procedure MTRANS (M : LIST): LIST;
{Matrix transpose. M is a matrix. The transposed matrix is returned. }

procedure VEL (a, n : LIST): LIST;
{Vector elements. A vector of length n with elements a is returned. }

procedure MFILL (M, m, n: LIST): LIST;
{Matrix fill. M is an upper triangular matrix. A (m x n) matrix with zeros in the lower triangular part is returned. }

procedure MRANG (U: LIST): LIST;
{Matrix rang. U is an upper triangular matrix from a LU-decomposition. The rang of U is returned. }

procedure RNMHILBERT (m, n : LIST): LIST;
{Rational number matrix Hilbert. m, n integer. A (m x n) rational number Hilbert matrix is returned. }

procedure RNUM (m, n : LIST): LIST;
{Rational number unit matrix. m, n integer. A (m x n) rational number unit matrix is returned. }

procedure RNVWRITE (A, s : LIST);
{Rational number vector write. A is a rational number vector. A is written to the output stream. The rational numbers are written as rational numbers if s = -1, as decimal approximations, with s decimal digits if s ¿= 0 or in floating point format if s ¡ -1. }

procedure RNVREAD (): LIST;
{Rational number vector read. A rational number vector is read from the input stream, and returned. }

procedure RNMWRITE (A, s : LIST);
{Rational number matrix write. A is a rational number matrix. A is written to the output stream. The rational numbers are written as rational numbers if s = -1, as decimal approximations, with s decimal digits if s ¿= 0 or in floating point format if s ¡ -1. }

procedure RNMREAD (): LIST;
{Rational number matrix read. A rational number matrix is read from the input stream, and returned. }

procedure RNVFIV (A : LIST): LIST;
{Rational number vector from integer vector. A is an integer vector. A rational number vector with denominators 1 and nominators equal to the elements of A is returned. }

procedure RNMFIM (M : LIST): LIST;
{Rational number matrix from integer matrix. A is an integer matrix. A rational number matrix with denominators 1 and nominators equal to the elements of A is returned. }

procedure RNVDIF (A, B : LIST): LIST;
{Rational number vector difference. A and B are rational number vectors. The rational number vector C = A - B is returned. }

procedure RNVQ (A, B : LIST): LIST;
{Rational number vector quotient. A and B are rational number vectors. The rational number vector C = A / FIRST(B) is returned. }

procedure RNVQF (A : LIST): LIST;
{Rational number vector quotient. A is a rational number vector. The rational number vector C = A / FIRST(A) is returned. }

procedure RNVVSUM (A, B : LIST): LIST;
{Rational number vector vector sum. A and B are rational number vectors. A rational number vector C = A + B is returned. }

procedure RNVSVSUM (A, B : LIST): LIST;
{Rational number vector scalar sum. A and B are rational number vectors. A rational number vector C = A + FIRST(B) is returned. }

procedure RNVSSUM (A : LIST): LIST;
{Rational number vector scalar sum. A is a rational number vector. A rational number C = a1 + a2 + ... + an is returned. }

procedure RNVSVPROD (A, B : LIST): LIST;
{Rational number vector scalar vector product. A and B are rational number vectors. A rational number vector C = (a1*FIRST(B), ..., an*FIRST(B)) is returned. }

procedure RNVVPROD (A, B : LIST): LIST;
{Rational number vector vector product. A and B are rational number vectors. A rational number vector C = (a1*b1, ..., an*bn) is returned. }

procedure RNVSPROD (A, B : LIST): LIST;
{Rational number vector scalar product. A and B are rational number vectors. A rational number C = a1*b1 + ... + an*bn is returned. }

procedure RNVMAX (M : LIST): LIST;
{Rational number vector maximum norm. M is a rational number vector. A rational number a = maximum absolute value M(i) is returned. }

procedure RNVLC (a, A, b, B : LIST): LIST;
{Rational number vector linear combination. A and B are rational number vectors. a and b are rational numbers. A rational number vector C = a*A + b*B is returned. }

procedure RNSVPROD (a, A : LIST): LIST;
{Rational number vector product with scalar. A is a rational number vector. a is a rational number. A rational number vector C = a*A is returned. }

procedure RNMSUM (A, B : LIST): LIST;
{Rational number matrix sum. A and B are rational number matrices. A rational number matrix C = A + B is returned. }

procedure RNMDIF (A, B : LIST): LIST;
{Rational number matrix difference. A and B are rational number matrices. A rational number matrix C = A - B is returned. }

procedure RNMPROD (A, B : LIST): LIST;
{Rational number matrix product. A and B are rational number matrices. A rational number matrix C = A * B is returned, if the number of coloums of A is equal to the number of rows of B, otherwise the empty matrix is returned. }

procedure RNSMPROD (A, B : LIST): LIST;
{Rational number scalar and matrix product. A is a rational number matrix. B is a rational number. A rational number matrix C = A * B is returned. }

procedure RNMMAX (M : LIST): LIST;
{Rational number matrix maximum norm. M is a rational number matrix. A rational number a = maximum absolute value M(i,j) is returned. }

procedure RNMGE (M : LIST): LIST;
{Rational number matrix Gaussian elimination. M is a (n x m) rational number matrix. A (n

x m) rational number matrix resulting from Gaussian elimination is returned. RNMGELUD is called. }

procedure RNMDET (M : LIST): LIST;
{Rational number matrix determinant, using Gaussian elimination. M is a rational number matrix. The determinant of M is returned. }

procedure RNMDETL (M : LIST): LIST;
{Rational number matrix determinant, using Laplace expansion. M is a rational number matrix. The determinant of M is returned. }

procedure RNMGELUD (M : LIST; VAR L, U: LIST);
{Rational number matrix Gaussian elimination LU-decomposition. M is a rational number matrix represented rowwise. L is a lower triangular rational number matrix represented columnwise. U is an upper triangular rational number matrix represented rowwise. M = L * U for appropriate modifications of L and U. The pivot operations are also recorded in L. }

procedure RNMLT (L, b : LIST): LIST;
{Rational matrix lower triangular matrix transformation. L is a lower triangular rational number matrix represented columnwise as generated by RNMGELUD. b is a rational number vector. A rational number vector u = L * b is returned, such that if M * x = b and M = L * U, then U * x = u. }

procedure RNMUT (U, b : LIST): LIST;
{Rational matrix upper triangular matrix transformation. U is an upper triangular rational number matrix represented rowwise as generated by RNMGELUD. b is a rational number vector b = L * b' as generated by RNMLT. A rational number vector x, such that U * x = b is returned. If no such x exists, then an empty vector is returned. If more than one such x exists, then for free x(i), x(i) = 0 is taken. }

procedure RNMSDS (L, U, b : LIST): LIST;
{Rational number matrix solve decomposed system. L is a lower triangular rational number matrix represented columnwise, U is an upper triangular rational number matrix represented rowwise. L and U as generated by RNMGELUD. If M = L * U, then a rational number vector x, such that M * x = b is returned. If no such x exists, then an empty vector is returned. If more than one such x exists, then for free x(i), x(i) = 0 is taken. }

procedure RNMINV (A : LIST): LIST;
{Rational number matrix inversion. A is a rational number matrix represented rowwise. If it exists, the inverse matrix of A is returned, otherwise an empty matrix is returned. }

procedure RNMUNS (U : LIST): LIST;
{Rational number matrix upper triangular matrix solution null space. U is an upper triangular rational number matrix represented rowwise as generated by RNMGELUD. A matrix X of linear independent rational number vectors x is returned, such that for each x in X, U * x = 0 holds. If only x = 0 satisfies the condition U * x = 0, then the matrix X is empty. }

8.6 Noether Polynomial System

procedure NOEINF ();
{Noether Polynomial System Information.}

procedure NOENSP (PG: LIST);
{Noether Number of Noetherian Base Polynomials. The number of noetherian base polynomials is computed.}

procedure NOEL32 (M, K: GAMMAINT): LIST;
{Noether SK Polynomial Computation. The result of the procedure is a respresentation of the polynomial S_K w.r.t the polyomials S_M, ..., S_2, S_1 over the rational numbers.}

procedure MERGE (FLAG: BOOLEAN; BASE1, POL1: LIST; VAR BASE2, POL2: LIST);
{Noether Merge of Base Polynomials. The procedure merges BASE1 and BASE2 into BASE2, makes an update of POL1 and POL2 with respect to the new base and return the sum (FLAG = TRUE) or the product (FLAG = FALSE) of POL1 and POL2.}

procedure NOESRT (POL: LIST): LIST;
{Noether Polynomial Sort. The polynomial POL in noetherian representation is sorted with respect to the lexicographical term order and the given term order.}

procedure NOEPOW (PG: LIST; K: GAMMAINT): LIST;
{Noether SK Power Sum Computation. The result of the procedure is the list of orbit polynomials and (z_1, z_2, ..., z_n)-terms of the polynomial S_K w.r.t. the permutation group PG.}

procedure NOEPSM (POL1, POL2: LIST): LIST;
{Noether Polynomial Sum. The sum of the polynomials POL1 and POL2 in noetherian representation is computed.}

procedure NOEPPR (POL1, POL2, TERM: LIST): LIST;
{Noether Polynomial Product. The product of the polynomials POL1 and POL2 in noetherian representation is computed und every non necessary z-term is removed.}

procedure NOEPIP (POL, FACT: LIST): LIST;
{Noether Polynomial Factor Multiplication. The product of the polynomial POL in noetherian representation and the rational number FACT is computed.}

procedure NOEPRM (POL, TERM: LIST): LIST;
{Noether Remove. All non necessary z-terms are removed from the polynomial POL.}

procedure NOERED (PG, POL: LIST; VAR BASE, BASEPOL, REMPOL: LIST);
{Noether G-Symmetric Polynomial Computation. The PG-symmetric polynomial BASEPOL, which is the PG-symmetric polynomial reconstruction w.r.t. the base polynomials BASE, and the remainder polynomial REMPOL are computed from the polynomial POL w.r.t. the permutation group over the rational numbers after the method of E. Noether.}

8.7 SAC Linear Diophantine Equation System

procedure LDSMKB (A,BL: LIST; VAR XLS,N: LIST);
{Linear diophantine system solution, modified Kannan and Bachem algorithm. A is an m by n integral matrix. A is represented column-wise. b is an integral m-vector. If the diophantine system A*x=b is consistent, then xs is a particular solution and N is a list of basis vectors of the solution module of A*x=0. Otherwise, xs and N are null lists. A and b are modified.}

procedure LDSSBR (A,BL: LIST; VAR XLS,N: LIST);
{Linear diophantine system solution, based on Rosser ideas. A is an m by n integral matrix. A is represented column-wise. b is an integral m-vector. If the diophantine system A*x=b is consistent, then xs is a particular solution and N is a list of basis vectors of the solution module of A*x=0. Otherwise, xs and N are null lists. A and b are modified.}

procedure MAIPDE (RL,M: LIST): LIST;
{Matrix of integral polynomials determinant, exact division algorithm. M is a square matrix of integral polynomials in r variables, r ge 0, represented as a list. D is the determinant of M.}

procedure MAIPDM (RL,M: LIST): LIST;
{Matrix of integral polynomials determinant, modular algorithm. M is a square matrix of integral polynomials in r variables, r non-negative. D is the determinant of M.}

procedure MAIPHM (RL,ML,A: LIST): LIST;
{Matrix of integral polynomials homomorphism. A is a matrix of integral polynomials in r variables, r non-negative. m is a positive beta-integer. B is the matrix B(i,j) of polynomials in r

variables over Z sub m such that B(i,j)=H(m)(A(i,j)).}

procedure MAIPP (RL,A,B: LIST): LIST;
{Matrix of integral polynomials product. A and B are matrices of integral polynomials in r variables, r ge 0, for which the matrix product A*B is defined. C=A*B.}

procedure MIAIM (A: LIST): LIST;
{Matrix of integers, adjoin identity matrix, A is an m by n matrix of integers. A is represented column-wise. B is the matrix obtained by adjoining an n by n identity matrix to the bottom of A. A is modified.}

procedure MICINS (A,V: LIST): LIST;
{Matrix of integers column insertion. A is an m by n integral matrix represented by the list (a(1),a(2), ...,a(n)), where a(i) is the list (a(1,i), ...,a(m,i)) representing column i of A and a(1,1) ge a(1,2) ge ... ge a(1,m). V=(v(1), ...,v(m)) is an integral vector with v(1) lt a(1,1). Let i be the largest integer such that a(1,i) ge v(1). Then B is the matrix represented by the list (a(1), ..., a(i),V,a(i+1), ...,a(n)). A is modified.}

procedure MICS (A: LIST): LIST;
{Matrix of integers column sort. A is an integral matrix with non- negative elements in first row. A is represented column-wise. B is an integral matrix obtained by sorting columns of A such that elements of the first row are in descending order. A is modified.}

procedure MINNCT (A: LIST): LIST;
{Matrix of integers, non-negative column transformation. A=(a(i,j)) is an m by n integral matrix. A is represented column-wise. B=(b(i,j)) is the m by n integral matrix with b(i,j)=a(i,j) if a(1,j) ge 0 and b(i,j)=-a(i,j) if a(1,j) lt 0. A is modified.}

procedure MMDDET (PL,M: LIST): LIST;
{Matrix of modular digits determinant. p is a prime beta-integer. M is a square matrix over GF(p), represented as a list. d is the determinant of M.}

procedure MMDNSB (PL,M: LIST): LIST;
{Matrix of modular digits null space basis. p is a prime beta- integer. M is an m by n matrix over Z sub p. B is a list (b(1), ..., b(r)) representing a basis for the null space of M, consisting of all x such that M*x=0. r is the dimension of the null space. B=() if the null space of M is 0. Each b(i) is a list representing an m-vector. M is modified. Alternatively, if M represents a matrix by columns, then B is a basis for the null space consisting of all x such that x*M=0.}

procedure MMPDMA (RL,PL,M: LIST): LIST;
{Matrix of modular polynomials determinant, modular algorithm. M is a square matrix of modular polynomials in r variables over Z sub p, r non-negative, p a prime beta-integer. D is the determinant of M.}

procedure MMPEV (RL,ML,A,KL,AL: LIST): LIST;
{Matrix of modular polynomials evaluation. A is a matrix of polynomials in r variables over Z sub m, m a positive beta-integer. 1 le k le r and a is an element of Z sub m. B is the matrix of polynomials b(i,j) where b(i,j)(x(1), ...,x(k-1),x(k+1), ...,x(r))= a(i,j)(x(1), ...,x(k-1),a,x(k+1), ...,x(r)).}

procedure VIAZ (A,NL: LIST): LIST;
{Vector of integers, adjoin zeros. A is the vector (a(1), ...,a(m)). n is a non-negative beta-integer. B is the vector (a(1), ...,a(m), 0, ...,0) of m+n components. A is modified.}

procedure VIDIF (A,B: LIST): LIST;
{Vector of integers difference. A and B are vectors in Z sup n. C=A-B.}

procedure VIERED (U,V,IL: LIST): LIST;
{Vector of integers, element reduction. U=(u(1), ...,u(n)) and V=(v(1), ...,v(n)) are integral n-vectors. 1 le i le n. v(i) ne 0. W=U-q*V, where q=INTEGER(u(i)/v(i)).}

procedure VILCOM (AL,BL,A,B: LIST): LIST;
{Vector of integers linear combination. a and b are integers. A and B are integral vectors in Z sup n. C=a*A+b*B.}

procedure VINEG (A: LIST): LIST;
{Vector of integers negation. A is an integral vector. B=-A.}

procedure VISPR (AL,A: LIST): LIST;
{Vector of integers scalar product. a is an integer. A is an integral vector. C=a*A.}

procedure VISUM (A,B: LIST): LIST;
{Vector of integers sum. A and B are vectors in Z sup n. C=A+B.}

procedure VIUT (U,V,IL: LIST; VAR UP,VP: LIST);
{Vector of integers, unimodular transformation. U=(u(1), ...,u(n)) and V=(v(1), ...,v(n)) are vectors in Z sup n with u(i) ne 0. (UP,VP)=(U,V)*K where K is a unimodular matrix, depending on u(i) and v(i), whose elements are obtained from IDEGCD.}

8.8 Substitution Group Polynomial System

procedure SUBINF ();
{Substitution Group Polynomial System Information. }

procedure SUBSGR (M: GAMMAINT): LIST;
{Substitution Group Read. Input of producing elements of a substitution group with M variables. }

procedure SUBSGW (SG: LIST);
{Substitution Group Write. Output of the substitution group SG. }

procedure SUBORD (SG: LIST): GAMMAINT;
{Substitution Group Order. The order of the substitution group SG is computed. }

procedure SUBORP (SG, POL: LIST): LIST;
{Substitution Group Orbit Polynomial. The orbit polynomial of the polynom POL is computed with respect to the substitution group SG. }

procedure SUBSYM (SG, POL: LIST): GAMMAINT;
{G-symmetric Polynomial Symmetric Check. The Procedure returns 1, if POL is SG-symmetric. Otherwise 0. }

procedure SUBOPL (SG, ML: LIST): LIST;
{G-symmetric Orbit Polynomial List. The SG-symmetric orbit polynomial list is calculated from the monomial list ML and the substitution group SG. }

procedure SUBCHK (SG, BASE, POL: LIST): LIST;
{G-Symmetric Polynomial Check. The original polynomial is computed from the polynomial POL and the SG-symmetric base polynomials BASE with respect to the substitution group PG. }

procedure SUBPOW (SG: LIST; K: GAMMAINT): LIST;
{Noether SK Power Sum Computation for Substitution Groups. The result of the procedure is the list of orbit polynomials and (z_1, z_2, ..., z_n)-terms of the polynomial S_K w.r.t. the substitution group PG. }

procedure SUBRED (SG, POL: LIST; VAR BASE, BASEPOL, REMPOL: LIST);
{Noether G-Symmetric Polynomial Computation for Substitution Groups. The SG-symmetric polynomial BASEPOL, which is the SG-symmetric polynomial reconstruction w.r.t. the base polynomials BASE, and the remainder polynomial REMPOL are computed from the polynomial POL w.r.t. the substitution group SG over the rational numbers after the method of E. Noether.}

8.9 Symmetric Functions

procedure DIRPSR (Q,PL: LIST; VAR P1,P2: LIST);
{Distributive rational polynomial symmetric function reduction. Q is a list of the rl elementary symmetric functions in rl variables. pl is reduced modulo Q to p2, the reduction relation is p1. }

procedure DIRPSE (Q,U: LIST; VAR PL,V: LIST);
{Distributive rational polynomial symm. function exponent reduction. Q is a list of the rl elementary symmetric functions in rl variables. pl is a product of elemenatry symmetric polynomials such that head term pl = u. v is the exponent vector of the product. }

procedure DIRPES (RL: LIST): LIST;
{Distributive rational polynomial elementary symmetric functions. Q is a list of the rl elementary symmetric functions in rl variables. }

procedure EVASC (U: LIST): LIST;
{Exponent vector ascending. U is an exponent vector of length rl, U=(u1, ... ,url). tl = 1 if u1 le ... le url, else tl = 0. }

8.10 Syzygy Functions

procedure SPC (P1, P2 : LIST; VAR SPFL, SP : LIST);
{S-Polynomial with Coefficients. Berechnet das S-Pol SP von P1 und P2, und speichert die zugehoerigen Faktoren Uij und Vij in dieser Reihenfolge unter SPFL ab. }

procedure NLSPC (P1, P2 : LIST; VAR SPFL, SP, T : LIST);
{Non-Commutative S-Polynomial with Coefficients. Berechnet nicht- kommutative das S-Pol SP von P1 und P2, und speichert die zugehoerigen Faktoren Uij und Vij in dieser Reihenfolge unter SPFL ab. }

procedure SPCGB (GB : LIST; VAR SPFL, SPL : LIST);
{S-Polynomials with Coefficients for Groebner Base. Berechnet die S-Polynome aller Polynome in GB und speichert diese in SPL ab. Die zugehoerigen Faktoren Uij und Vij aller S-Polynome werden in dieser Reihenfolge unter SPFL abgelegt. }

procedure NLSPCGB (GB : LIST; VAR SPFL, SPL, T : LIST);
{Non-Commutative S-Polynomials with Coefficients for Groebner Base. Berechnet die nicht-kommutativen S-Polynome aller Polynome in GB und speichert diese in SPL ab. Die zugehoerigen Faktoren Uij und Vij aller S-Polynome werden in dieser Reihenfolge unter SPFL abgelegt. }

procedure SPCEGB (GB, L : LIST; VAR SPFL, SPL : LIST);
{S-Polynomials with Coefficients and Exponenetvector-Check for Groebner Base. Berechnet die S-Polynome aller Polynome in GB unter Beruecksichtigung der Gleichheit der fuehrenden L Stellen des Exponenetenvektors der HT, und speichert diese in SPL ab. Die zugehoerigen Faktoren Uij und Vij aller S-Polynome werden in dieser Reihenfolge unter SPFL abgelegt. Konnte ein S-Polynom nicht gebildet werden, dann wird an die Liste SPFL als Steuerbit eine Null angehaengt. }

procedure NLSPCEGB (GB, L : LIST; VAR SPFL, SPL, T : LIST);
{Non-Commutative S-Polynomials with Coefficients and Exponenetvector-Check for Groebner Base. Berechnet die nicht-kommutativen S-Polynome aller Polynome in GB unter Beruecksichtigung der Gleichheit der fuehrenden L Stellen des Exponenetenvektors der HT, und speichert diese in SPL ab. Die zugehoerigen Faktoren Uij und Vij aller S-Polynome werden in dieser Reihenfolge unter SPFL abgelegt. Konnte ein S-Polynom nicht gebildet werden, dann wird an die Liste SPFL als Steuerbit eine Null angehaengt. }

procedure RCSP (GB, SPL : LIST): LIST;
{Reduction Chain of S-Polynomials. Fuer alle S-Polynome in SPL werden diejenigen Faktoren

gespeichert, die noetig sind, um ein S-Polynom bzgl. der Groebner Basis GB zu Null zu reduzieren. Bsp.: Sei GB = (G1, G2, G3) und gelte SPi - P1 G1 - P2 G2 - P3 G1 = 0. Dann hat die i-te Zeile der Ergebnismatrix die Gestalt: (P1+P3, P2, 0). }

procedure RCSPR (PL : LIST; VAR SP : LIST): LIST;
{Reduction Chain of S-Polynomials with Remainder. Fuer das S-Polynome SP werden diejenigen Faktoren gespeichert, die noetig sind, um dieses S-Polynom bzgl. der Polynome in PL zu Null zu reduzieren. Konnte bei einem Durchlauf nicht mehr reduziert werden, dann bricht das Verfahren mit den bereits errechneten Werten ab. Bsp.: Sei PL = (P1, P2, P3) und gelte SP - F1 P1 - F2 P2 - F3 P1 = 0. Dann hat die Ergebniszeile die Gestalt: (F1+F3, F2, 0). }

procedure NLRCSPR (PL : LIST; VAR SP, T : LIST): LIST;
{Reduction Chain of S-Polynomials with Remainder. Fuer das S-Polynome SP werden diejenigen Faktoren gespeichert, die noetig sind, um dieses S-Polynom bzgl. der Polynome in PL zu Null zu reduzieren. Konnte bei einem Durchlauf nicht mehr reduziert werden, dann bricht das Verfahren mit den bereits errechneten Werten ab. Bsp.: Sei PL = (P1, P2, P3) und gelte SP - F1 P1 - F2 P2 - F3 P1 = 0. Dann hat die Ergebniszeile die Gestalt: (F1+F3, F2, 0). }

procedure NLRCSP (GB, SPL : LIST; VAR T : LIST): LIST;
{Non-Commutative Reduction Chain of S-Polynomials. Fuer alle S-Polynome in SPL werden diejenigen Faktoren gespeichert, die noetig sind, um eine S-Polynom bzgl. der Groebner Basis GB und der Kommutatorrelationen T zu Null zu reduzieren. Bsp.: Sei GB = (G1, G2, G3) und gelte SPi - P1 G1 - P2 G2 - P3 G1 = 0. Dann hat die i-te Zeile der Ergebnismatrix die Gestalt: (P1+P3, P2, 0). }

procedure SYGB (SPFL, SPAK : LIST): LIST;
{Syzygy for Groebner Base. Berechnet wird aufgrund bereits erzeugter Faktorenliste SPFL und den S-Polynom Ableitungsketten SPAK ein Loesungsmodulgenerator fuer eine homogene Gleichung, wobei die Polynome dieser Gleichung eine Groebner Basis bilden. }

procedure SYGBE (SPFL, SPAK : LIST): LIST;
{Syzygy for Groebner Base with Exponent Vector. Berechnet wird aufgrund bereits erzeugter Faktorenliste SPFL und den S-Polynom Ableitungsketten SPAK ein Loesungsmodulgenerator fuer eine homogene Gleichung, wobei die Polynome dieser Gleichung eine Groebner Basis bilden. Hier wird das bei SPCEGB in die Faktorliste SPFL eingetragene Steuerbit 0 verwertet, das besagt, dass sofort ein neuer Schleifendurchlauf beginnen soll. }

procedure MMULT (SY1, GBTM : LIST): LIST;
{Matrix Multiplication. Das Produkt der Matrizen SY1 * GBTM ergibt die Loesungsmatrix. }

procedure NLMMULT (SY1, GBTM : LIST; VAR T : LIST): LIST;
{Non-Commutative Matrix Multiplication. Das nicht-kommutative Produkt der Matrizen SY1 * GBTM ergibt die Loesungsmatrix. }

procedure BGFUP (P1, P2, SP, SPN, SPFL, GB, SPAK, GBTM : LIST): LIST;
{Base Generators Factor Update. Fuer das zu einer Groebner Basis GB neu hinzugenommene normierte Polynom SPN werden dessen Abhaengigkeiten zu den Grundpolynomen berechnet. Dazu werden die Polynome P1 und P2, aus denen SP gebildet wurde, die zwei Faktoren des S-Polynoms in SPFL, die Ableitungskette des urspruenglichen S-Polynoms SPAK und die bereits bestehende Faktormatrix GBTM verwendet. }

procedure NLBGFUP (P1, P2, SP, SPN, SPFL, GB, SPAK, GBTM : LIST;
VAR T : LIST): LIST; {Non-Commutative Base Generators Factor Update. Fuer das zu einer Groebner Basis GB neu hinzugenommene normierte Polynom SPN werden dessen Abhaengigkeiten zu den Grundpolynomen berechnet. Dazu werden die Polynome P1 und P2, aus denen SP mit nicht-kommutativer Multiplikation gebildet wurde, die zwei Faktoren des S-Polynoms in SPFL und die bereits bestehende Faktormatrix GBTM verwendet. }

procedure DGBRED (GB, GBTM : LIST; VAR SY : LIST): LIST;
{Discrete Groebner Base Reduction. Konnte ein Polynom P aus GB bzgl. GB ohne P nicht zu

Null reduziert werden, dann verbleibt es im Ursprungszustand in der Groebner Basis GB. Hat die so reduzierte GB nur noch ein Polynom, dann wird die Syzygie abhaengig von der Faktormatrix GBTM berechnet. }

procedure NLDGBRED (GB, GBTM : LIST; VAR SY, T : LIST): LIST;
{Non-Commutative Discrete Groebner Base Reduction. Konnte ein Polynom P aus GB bzgl. GB ohne P nicht zu Null reduziert werden, dann verbleibt es im Ursprungszustand in der Groebner Basis GB. Verwendung findet hier die Bildung der Linksnormalform mit DINLNF. Hat die so reduzierte GB nur noch ein Polynom, dann wird die Syzygie abhaengig von der Faktormatrix GBTM berechnet. }

procedure SYONP (GB1, GB2, GBTM : LIST): LIST;
{Syzygy for old Polynomials by new Polynomials. Berechnet einen Loesungsmodulgenerator ausgehend von der Groebner Basis GB1 unter Zuhilfenahme der alten Polynome und der Faktormatrix GBTM. GB2 ist die eventuell reduzierte Groebner Basis. }

procedure NLSYONP (GB1, GB2, GBTM : LIST; VAR T : LIST): LIST;
{Syzygy for one Polynomial. Berechnet einen Loesungsmodulgenerator fuer das Polynom P unter Zuhilfenahme der Faktormatrix GBTM. }

procedure DINPQ (P1, P2 : LIST; VAR T : LIST): LIST;
{Distributive non-commutative Polynomial Quotient. Berechnet wird der nicht-kommutaive Quotient P1/P2=P3; }

procedure PLMULT (SY, PL : LIST): LIST;
{Polynomial List Multiplication. Multipliziert die Polynome der Listen SY und PL komponentenweise, und addiert die erhaltenen Ergebnisse. }

procedure NLPLMULT (SY, PL : LIST; VAR T : LIST): LIST;
{Non-Commutative Polynomial List Multiplication. Multipliziert die Polynome der Listen SY und PL komponentenweise, und addiert die erhaltenen Ergebnisse. }

8.11 Syzygy Groebner Base

procedure MGB (PM, SANZ : LIST): LIST;
{Modul Groebner Base. Berechnet wird die Modul Groebner Basis fuer die Polynommatrix PM. Das Bit SANZ steuert die Anzeigeart. }

procedure NLMGB (PM, SANZ : LIST; VAR T : LIST): LIST;
{Non-Commutative Modul Groebner Base. Berechnet wird die Modul Groebner Basis fuer die Polynommatrix PM. Das Bit SANZ steuert die Anzeigeart. }

procedure GBE (PL, SANZ, L : LIST): LIST;
{Groebner Base with Exponent Vector Check. Berechnung der Groebner Basis von PL unter Berueckksichtigung des Exponentenvektors der Hoechsten Terme. S-Polynome werden nur bei solchen Polynomen gebildet, deren HT-Exponen- tenvektor in den ersten L Stellen uebereinstimmt. }

procedure GBF (PL, SANZ: LIST; VAR GBTM : LIST): LIST;
{Groebner Base with Factors. Groebner Basis Berechnung mit Faktoren. Waehrend des Programmdurchlaufs wird die "Entstehungsgeschichte" der Groebner Basis von PL dokumentiert, d.h. jedes neu hinzugenommene Basispolynom wird dargestellt durch Faktoren GBTM bezogen auf die Ausgangspolynome. }

procedure GBEF (PL, SANZ, L : LIST; VAR GBTM : LIST): LIST;
{Groebner Base with Exponent Vector Check and Factors. Kombination der Eigenschaften der Funktionen GBE und GBF. }

procedure NLGBE (PL, SANZ, L : LIST; VAR T : LIST): LIST;
{Non-Commutative Groebner Base with Exponent Vector Check. Berechnung der Groebner Basis von PL unter Beruecksichtigung des Exponentenvektors der Hoechsten Terme. S-Polynome werden nur bei solchen Polynomen gebildet, deren HT-Exponentenvektor in den ersten L Stellen uebereinstimmt. }

procedure NLGBF (PL, SANZ: LIST; VAR GBTM, T: LIST): LIST;
{Non-Commutative Groebner Base with Factors. Groebner Basis Berechnung mit Faktoren. W"ahrend des Programmdurchlaufs wird die "Entstehungsgeschichte" der Groebner Basis von PL dokumentiert, d.h. jedes neu hinzugenommene Basispolynom wird dargestellt durch Faktoren GBTM bezogen auf die Ausgangspolynome. }

procedure NLGBEF (PL, SANZ, L : LIST; VAR GBTM, T : LIST): LIST;
{Non-Commutative Groebner Base with Exponent Vector Check and Factors. Kombination der Eigenschaften der Funktionen NGBE und NGBF. }

8.12 Syzygy Utility Programs

procedure ALFA (L : LIST);
{Automatic Linear Form Adaption. Fuehrt bei einer gegebenen Linearform von EVORD L neue Variablen mit Graden groesser als der hoechste Grad der EVORD-Polynome ein. }

procedure ALFRA (L : LIST);
{Automatic Linear Form Readaption. Gegenstueck zu ALFA. Reduziert bei gegebener Linearform von EVORD die fuehrenden L Polynome. }

procedure ADDPPOS (PL, P, POS : LIST): LIST;
{Add Polynomial P at Position. Gegeben ist eine Polynomliste PL. Dann wird P an der POS-ten Position dieser Liste zu dem dort vorhandenen Polynom addiert. }

procedure PLWR (PL, VL : LIST);
{Polynomial List Write. Am Bildschirm wird die Polynomliste PL bzgl. der Variablenliste VL ausgegeben. }

procedure PMWR (PM, VL : LIST);
{Polynomial Matrix Write. Am Bildschirm wird die Polynommatrix PM (Liste ueber Listen) bzgl. der Variablenliste VL ausgegeben. }

procedure APP0 (PM : LIST): LIST;
{Append 0. Haengt an jede Polynomliste von PM das Nullpolynom an. }

procedure ADDLAST (P, PL : LIST): LIST;
{Add last Polynomial. Addiert das Polynom P zum letzten Polynom der Polynomliste PL. }

procedure POS (P, PL : LIST): LIST;
{Position of Polynomial. Bestimmt die Position des Polynoms P in der Polynomliste PL. }

procedure POL (PL, POS : LIST): LIST;
{Polynomial at Position. Bestimmt das Polynom an der POS-ten Stelle in der Polynomliste PL. }

procedure GENPOSV (GB, GBR : LIST): LIST;
{Generate Postion Vector. Gegeben ist eine Groebner Basis GB und die dazu- gehoerige diskret reduzierte Groebner Basis GBR. Bestimmt wird nun ein Vektor mit Nullen und Einsen, bei dem die Einsen an der Position stehen, an der ein Polynom aus GB nicht ganz zu Null reduziert werden konnte. Zusaetzlich bleiben immer mindestens zwei Polynome aus GB uebrig. Bsp.: GB = (P1, P2, P3), GBR = (P2, P3), dann ist POSV = (0, 1, 1). }

procedure INSPOSV (PM, POSV : LIST): LIST;
{Insert Position Vector. Fuegt bei der Polynommatrix PM in jede Polynomliste PL Nullen an

den Stellen ein, an denen bei POSV auch Nullen stehen. Bsp.: Sei eine Polynomliste PLi = (P1, P2, P3), und sei POSV = (0, 1, 0, 1, 1, 0, 0). Dann wird als neue Polynomliste (0, P1, 0, P2, P3, 0, 0) an die Matrix zurueckgegeben. }

procedure EXPPL (P, GB : LIST): LIST;
{Exclude P from GB. Loescht das Polynom P aus der Polynomliste GB. }

procedure EX0PL (PL : LIST): LIST;
{Exclude 0 from PL. Loescht alle Nullen aus der Polynomliste PL. }

procedure EVF (EV, L : LIST): LIST;
{Exponent Vector First. Liefert die ersten L Stellen des Exponentenvektors EV zurueck. Bsp.: Sei EV = (4, 3, 0, 1, 9), und sei L = 3, dann wird (4, 3, 0) zurueckgegeben. }

procedure EVR (PM, L : LIST): LIST;
{Exponent Vector Reduction. Bei der Polynommatrix PM wird jeder Exponentenvektor jeden Polynoms um die ersten L Stellen gekuerzt. }

procedure MREAD (VL : LIST): LIST;
{Matrix Read. Liest eine Polynommatrix entsprechend der Variablenliste VL und der gegebenen Einheit (Textdatei, "Bildschirm") ein. }

procedure NMREAD (VL, T : LIST): LIST;
{Non-Commutative Matrix Read. Liest eine Polynommatrix mit nicht- kommutativen Polynomen entsprechend der Variablenliste VL, der Relations- matrix T und der gegebenen Einheit (Text-datei, "Bildschirm") ein. }

procedure TA (L : LIST; T : LIST): LIST;
{T Adaption. Die Exponentenvektoren jeden Polynoms in der Polynomliste T werden um L Stellen erweitert. Bsp.: Sei EV = (2, 1, 3) und L = 2, dann wird EV zu (0, 0, 2, 1, 3). }

procedure TR (L : LIST; T : LIST): LIST;
{T Readaption. Die Exponentenvektoren jeden Polynoms in der Polynomliste T werden um L Stellen gekuerzt. Bsp.: Sei EV = (0, 0, 2, 1, 3) und L = 2, dann wird EV zu (2, 1, 3). }

procedure NEXTPAIR (VAR P1, P2, PPL : LIST);
{Next Pair of Polynomials. Bestimmt aus der Polynompaarliste PPL das naechste Paar P1, P2 von Polynomen. Gleichzeitig wird dieses Paar aus der Liste entfernt. Siehe Groebner Basis Algorithmen! }

procedure EVT (P1, P2, L : LIST): LIST;
{Exponent Vector Test. Ueberprueft, ob die Exponentenvektoren der HT der Polynome P1 und P2 in den fuehrenden L Stellen uebereinstimmen. Ist dies der Fall, dann wird die 1, ansonsten die 0 zurueckgegeben. }

procedure WRS1 (SZ, C1, C2, C3 : LIST);
{Write Situation. Ausgegben wird die CPU-Zeit minus eine Startzeit SZ, die Anzahl der H-Polynome C1, die Anzahl der S-Polynome C2 und die Anzahl der uebrigen Paare von Polynomen C3. }

procedure WRS2 (SZ, C1, TW1, C2, SPN, C3 : LIST);
{Write Situation. Ausgegben wird die CPU-Zeit minus eine Startzeit SZ, die Anzahl der H-Polynome C1 und das letzte H-Polynom, die Anzahl der S-Polynome C2 und das letzte S-Polynom, sowie die Anzahl der uebrigen Paare von Polynomen C3. }

procedure EVL (PM : LIST) : LIST;
{Exponent Vector Length. Bestimmt in einer Polynommatrix PM die Laenge des Exponenten-vektors des ersten Polynoms ungleich dem Nullpolynom. }

procedure NORMF (VAR PL, GBTM : LIST);
{Normative Factors. Berechnet eine Matrix GBTM, auf deren Hauptdiagonalen die Normfaktoren der Polynome der Polynomliste PL stehen. }

procedure GBTMRED (GBTM, POSV : LIST) : LIST;
{GBTM Reduction. Reduziert die Spalten von GBTM entsprechend der auftretenden Nullen in POSV. }

procedure MTPLV (PM : LIST; VAR L : LIST): LIST;
{Matrix to Polynomial List Vertical. Erzeugt eine Polynomliste PL derart, dass L (gleich Zeilenzahl) neue verschiedene Variablen mit der Matrix multipliziert werden (1. Variable mit der 1. Zeile, ...), und anschliessend die so erhaltene Matrix spaltenweise aufaddiert wird. }

procedure PLVTM (PL, L : LIST): LIST;
{Polynomial List Vertical To Matrix. Das Gegenstueck zu MTPLV. Beachtet werden mua hierbei nur, dass Anteile der Polynome aus der Polynomliste PL enstprechend der L neu eingefuehrten Variablen wieder in eine Matrix zerlegt werden, d.h. das erste Polynom der Polynomliste PL verteilt sich in der ersten Spalte, ... }

procedure MTPLH (PM : LIST; VAR L : LIST): LIST;
{Matrix to Polynomial List Horizontal. Erzeugt eine Polynomliste PL derart, dass L (gleich Spaltenzahl) neue verschiedene Variablen mit der Matrix multipliziert werden (1. Variable mit der 1. Spalte, ...), und anschlieaend die so erhaltene Matrix zeilenweise aufaddiert wird. }

procedure PLHTP (PL : LIST): LIST;
{Polynomial List Horizontal To Polynomial. Jedes Polynom aus PL wird mit einer neuen , zu den anderen verschiedenen, Variablen multipliziert und zu einem Polynom aufaddiert. }

procedure VMADD (PM : LIST): LIST;
{Vertical Matrix Addition. Erzeugt wird eine Polynomliste, deren Elemente aus den aufaddierten Spalten der Matrix PM gebildet wurden. }

8.13 Syzygy Main Programs

procedure SYHC (PM, SANZ, SRD : LIST): LIST;
{Syzygy for homogenous commutative system of equation. Berechnet die Syzygien fuer ein kommutatives Gleichungssystem. Das Bit SRD steuert eine moegliche Reduktion waehrend des Verfahrens. Das Bit SANZ regelt die Anzeigeart von Zwischenergebnissen. }

procedure HEQ (PL, SANZ, SRD : LIST): LIST;
{Homogenous Equation. Berechnet den Syzygienmodul fuer eine Gleichung mit den Polynomen aus PL. Das Bit SRD steuert eine moegliche Reduktion waehrend des Verfahrens. Das Bit SANZ regelt die Anzeigeart von Zwischenergebnissen.}

procedure HSEQ (PM, SANZ, SRD : LIST): LIST;
{Homogenous System of Equation. Berechnet den Syzygienmodul fuer das Gleichungssystem mit den Polynomen aus der Matrix PM. Das Bit SRD steuert eine moegliche Reduktion waehrend des Verfahrens. Das Bit SANZ regelt die Anzeigeart von Zwischenergebnissen.}

procedure SYTHC (SY, PM, VL : LIST);
{Syzygy Test for homogenous commutative Case. Testet, ob der berechnete Loesungsgenerator SY jede einzelne Gleichung von PM loest. Die Polynome der errechnten linken Seiten der Gleichungen werden ausgegeben. }

procedure SIC (PM, PL, SANZ, SRD : LIST): LIST;
{Special Solution for inhomogenous commutative system of equation. Berechnet eine spezielle Loesung fuer ein kommutatives Gleichungssystem PM. Die Polynome der rechten Seite stehen in der Liste PL. Das Bit SRD steuert eine moegliche Reduktion waehrend des Verfahrens. Das Bit SANZ regelt die Anzeigeart von Zwischenergebnissen.}

procedure IEQ (PL, P, SANZ, SRD : LIST): LIST;
{Special Solution for inhomogenous commutative equation. Berechnet eine spezielle Loesung fuer eine lineare Gleichung PL. Das Polynom der rechten Seite ist P. Das Bit SRD steuert

eine moegliche Reduktion waehrend des Verfahrens. Das Bit SANZ regelt die Anzeigeart von Zwischenergebnissen.}

procedure ISEQ (PM, PL, SANZ, SRD : LIST): LIST;
{Special Solution for inhomogenous commutative system of equation. Berechnet eine spezielle Loesung fuer ein kommutatives Gleichungssystem PM. Die Polynome der rechten Seite stehen in der Liste PL. Das Bit SRD steuert eine moegliche Reduktion waehrend des Verfahrens. Das Bit SANZ regelt die Anzeigeart von Zwischenergebnissen.}

procedure STIC (SY, PM, PL, VL : LIST);
{Solution Test for inhomogenous commutative Case. Testet, ob der berechnete Loesungsvektor SY jede einzelne Gleichung von PM loest. Die Polynome der errechnten linken Seiten werden von denen der rechten Seite abgezogen, und das Ergebnis dieser Differenz wird ausgegeben. }

procedure SYHNL (PM, SANZ, SRD, T : LIST): LIST;
{Syzygy for homogenous non-commutative system of equation. Berechnet die Syzygien fuer ein nicht-kommutatives Gleichungssystem. Das Bit SRD steuert eine moegliche Reduktion waehrend des Verfahrens. Das Bit SANZ regelt die Anzeigeart von Zwischenergebnissen. }

procedure NLHEQ (PL, SANZ, SRD, T : LIST): LIST;
{Non-Commutative Homogenous Equation. Berechnet den Syzygienmodul fuer eine Gleichung mit den Polynomen aus PL. Das Bit SRD steuert eine moegliche Reduktion waehrend des Verfahrens. Das Bit SANZ regelt die Anzeigeart von Zwischenergebnissen.}

procedure NLHSEQ (PM, SANZ, SRD, T : LIST): LIST;
{Non-Commutative Homogenous System of Equation. Berechnet den Syzygienmodul fuer das Gleichungssystem mit den Polynomen aus der Matrix PM. Das Bit SRD steuert eine moegliche Reduktion waehrend des Verfahrens. Das Bit SANZ regelt die Anzeigeart von Zwischenergebnissen.}

procedure SYTHNL (SY, PM, VL, T : LIST);
{Syzygy Test for homogenous non-commutative Case. Testet, ob der berechnete Loesungsgenerator SY jede einzelne Gleichung von PM loest. Die Polynome der errechnten linken Seiten der Gleichungen werden ausgegeben. }

procedure SINL (PM, PL, SANZ, SRD, T : LIST): LIST;
{Special Solution for inhomogenous non-commutative system of equation. Berechnet eine spezielle Loesung fuer ein nicht-kommutatives Gleichungs- system PM. Die Polynome der rechten Seite stehen in der Liste PL. Das Bit SRD steuert eine moegliche Reduktion waehrend des Verfahrens. Das Bit SANZ regelt die Anzeigeart von Zwischenergebnissen.}

procedure NLIEQ (PL, P, SANZ, SRD, T : LIST): LIST;
{Special Solution for inhomogenous non-commutative equation. Berechnet eine spezielle Loesung fuer eine lineare Gleichung PL. Das Polynom der rechten Seite ist P. Das Bit SRD steuert eine moegliche Reduktion waehrend des Verfahrens. Das Bit SANZ regelt die Anzeigeart von Zwischenergebnissen.}

procedure NLISEQ (PM, PL, SANZ, SRD, T : LIST): LIST;
{Special Solution for inhomogenous non-commutative system of equation. Berechnet eine spezielle Loesung fuer ein nicht-kommutatives Gleichungs- system PM. Die Polynome der rechten Seite stehen in der Liste PL. Das Bit SRD steuert eine moegliche Reduktion waehrend des Verfahrens. Das Bit SANZ regelt die Anzeigeart von Zwischenergebnissen.}

procedure STINL (SY, PM, PL, VL, T : LIST);
{Solution Test for inhomogenous non-commutative Case. Testet, ob der berechnete Loesungsvektor SY jede einzelne Gleichung von PM loest. Die Polynome der errechnten linken Seiten werden von denen der rechten Seite abgezogen, und das Ergebnis dieser Differenz wird ausgegeben. }

procedure OREC (P1, P2 : LIST; VAR P3, P4, T : LIST);
{Ore Condition. Fuer gegebene Polynome P1 und P2 werden Polynome P3 und P4 berechnet, sodass P3*P1 = P4*P2. Die Multiplikation * ist die nicht- kommutative Multiplikation. }

8.14 DIP Rational Extended Groebner Bases

procedure DIREGB (P, TF: LIST; VAR GB, GBM: LIST);
{Distributive rational polynomials extended groebner basis. }

Chapter 9

MAS Ring Algorithms

9.1 DIP Ideal Decomposition 0 System

procedure DIGFET (P,IL,JL: LIST): LIST;
{DIP G base successful extension test. P is a Groebner base of an ideal of dimension 0 in inverse lexicographical term ordering. i and j are indexes of variables where an field extension is required. t=1 if the extension was successful t=0 else. }

procedure DIGISM (P: LIST): LIST;
{DIP G base index search for extension multiple univariats. P is a Groebner base of dimension 0 in inverse lexicographical term ordering. I is a list of indexes of variables where an field extension is required or I=() if no field extension is neccessary. }

procedure DIGISR (P: LIST): LIST;
{DIP G base index search for extension reductas. P is a Groebner base of an ideal of dimension 0 in inverse lexicographical term ordering. I is a list of indexes of variables where an field extension is required or I=() if no field extension is neccessary. }

procedure DINTFE (T,IL,JL: LIST): LIST;
{DIP normalized tupel field extension. T is a normalized tupel of a zero set with a final Groebner base of dimension 0. i and j determine the variable indexes for the field extension. TP is a list of normalized tupels for the field extension for T. Trial values are used for the transcendent parameter. }

procedure DINTSR (T: LIST): LIST;
{DIP normalized tupel separation refinement. T is a list of normalized tupels with final Groebner base of dimension 0. TP is a list of normalized tupels for some field extensions for T. }

procedure DINTSS (T: LIST): LIST;
{DIP normalized tupel strong separation. T is a list of normalized tupels with final Groebner base of dimension 0. TP is a list of normalized tupels for some field extensions for T. }

procedure DINTZS (N: LIST): LIST;
{DIP nomalized tupels from system zero. N is a zero set. T is the list of nomalized tupels of N. }

procedure DIRGZS (VB,PB,W: LIST): LIST;
{Distributive rational Groebner base zero set. VB is a rest of a variable list. PB is a Groebner base. W is the total variable list. N is the zero set of P. }

procedure DIRLPD (A,VP: LIST): LIST;
{DIP rational polynomial ideal primary ideal decomposition. A is a non empty list of distributive rational polynomials representing a Groebner base. The polynomials in A have r variables. L=(l1,

...,ln) with li=(pi,ei,vpi,qi) i=1, ...,n where qi = ideal(pi**e,A) with A contained in qi and e maximal. Ideal(pi) is a prime ideal in at most r+1 variables. VPI is the variable list vor pi. }

procedure DIRLPW (A,V,L: LIST);
{DIP rational polynomial ideal primary ideal decomposition write. A is a non empty list of distributive rational polynomials representing a Groebner base. The polynomials in a have r variables. L=(l1, ...,ln) with li=(pi,eli,vpi,qi) i=1, ...,n where qi = ideal(pi)**e with A contained in qi and e maximal. Ideal(pi) is a prime ideal in at most r+1 variables. VPI is the variable list for pi. }

procedure DIRPDA (A,VP: LIST): LIST;
{DIP rational polynomial ideal primary ideal decomposition over Q(alpha). A is a non empty list of distributive rational polynomials representing a Groebner base. The polynomials in A have r variables. L=(l1, ...,ln) with li=(pi,ei,vpi,qi) i=1, ...,n where qi = ideal(pi**e,A) with A contained in qi and e maximal. Ideal(pi) is a prime ideal in at most r+1 variables. VPI is the variable list vor pi. }

procedure DITFZS (N: LIST): LIST;
{DIP tupel from zero set. N is a zero set. T is a list of tupels of the zero set. }

procedure DITSPL (T: LIST; VAR T0,T1: LIST);
{DIP zero set tupel split. T is a list of normalized tupels of a zero set. T0 is a list of normalized tupels of a zero set with a final Groebner base of a ideal of dimension 0. T1=T-T0. }

9.2 DIP Dimension

procedure DIGBZT (S: LIST): LIST;
{Distributive polynomial groebner base common zero test. S is a groebner basis. t = -1 or 0 if DIMENSION(S) eq -1 or 0, t = 1 if DIMENSION(S) ge 1. }

procedure DILDIM (G: LIST; VAR DL,S,M: LIST);
{Distributive polynomial list dimension. G is a list of distributive polynomials, a groebner base. d is the dimension of ideal(G). M is a maximal independend set of variables. S is a set of maximal independent sets of variables. }

procedure DIDIMS (G,S,U,M: LIST): LIST;
{Distributive polynomial dimension maximal independent set. G is a list of distributive rational polynomials, and a g-base. S is a maximal independent set of variables. U is a set of variables of unknown status. M and MP are lists of maximal independent sets of variables. }

procedure EVGBIT (S,G: LIST): LIST;
{Exponent vector groebner base intersection test. S is a set of variable indices. G is a groebner basis. t = 0 if intersection = () else t = 1. }

procedure USETCT (U,V: LIST): LIST;
{Unordered set containment test. U and V are unordered sets. t = 1 if U is contained in V else t = 0. }

procedure IXSUBS (V,I: LIST): LIST;
{Indexed subset. V is a list. I is an index list. The elements of V with index from I are put to VP. }

procedure DIDIMWR (DL,S,M,V: LIST);
{Distributive polynomial dimension write. d is the dimension of an ideal. M is a maximal independend set of variables. S is a set of maximal independent sets of variables. V is the variable list. }

9.3 DIP GCD

procedure DIRFAC (P: LIST): LIST;
{Distributive rational polynomial factorisation. P is a distributive rational polynomial. PP=((e1,P1), ...,(en,Pn)), where P=P1**e1+ ... +Pn**en. }

procedure IPLCM (RL,A,B: LIST): LIST;
{Integral polynomial least common multiple. A and B are integal polynomials. C=LCM(A,B), a nonnegative integral polynomial.}

9.4 DIP Ideal System

procedure DIPLDV (A,V: LIST): LIST;
{Distributive polynomial list dependency on variables. A is a list of distributive polynomials. V is the variable list. U is the variable list of variables with positive exponents in A. }

procedure DIRLCT (A,B: LIST): LIST;
{Distributive rational polynomial list ideal containment test. A and B are lists of distributive rational polynomials representing groebner bases. t = 1 if ideal(A) is contained in ideal(B), t = 0 else. }

procedure DIRLIP (PL,A,B: LIST): LIST;
{Distributive rational polynomial list ideal product. A and B are lists of distributive rational polynomials. C=GBASIS(p,A*B).}

procedure DIRLPI (A,P,VP: LIST): LIST;
{Distributive rational polynomial list primary ideal. A and P are non empty lists of distributive rational polynomials representing groebner bases. The polynomials in A have r variables. ideal(P) is a prime ideal in at most r+1 variables. VP is the variable list for P. QP=(P,e,VP,Q) where Q = ideal(P**e,A) with A contained in Q and e maximal. }

9.5 DIP Integral D-Groebner Bases

procedure DIIPELIMDGB (P : LIST) : LIST;
{Distributive integral polynomial eliminate D-groebner base. P is a list of non zero polynomials in distributive integral representation in r variables. ELIMDGB eliminates the polynominials with respect to the divisibility of the highest monominials. }

procedure DIIPTDR (P, lcmHT, pair : LIST): LIST;
{Distributive integral polynomial top-D-reduzibel. P is a list of non zero polynomials in distributive integral representation in r variables. pair is a pair two integral polynomials in distributive representation. lcmHT is the lcm of the highest terms of the two polynomials. TDR is a boolean value which equals 1, if g is top-D- reduzibel modulo P and 0 if not. }

procedure DIIPCPLMS1 (P : LIST) : LIST;
{Distributive integral polynomial list construct pairs list merge sort. P is a list of non zero polynomials in distributive integral representation in r variables. CPLMS1 sorts a constructed pairs list in the following ascending order: 1. lcm of the highest terms 2. lcm of the highest coefficients P will be changed. }

procedure DIIPLM1 (onestep, twostep : LIST) : LIST;
{Distributive integral polynomial list merge sort. P is a list of non zero polynomials in distributive integral representation in r variables. LM1 merges two (onestep, twostep) constructed pairs lists in the same manner as DIIPLCPLMS1. The lists onestep and twostep will be changed. }

procedure DIIPUCPL1 (P, g, Old : LIST) : LIST;
{Distributive polynomial D-update constructed pairs list. P is a list of integral polynomials in distributive representation. g ist a polynomial in distributive representation. Both are polynomials in r variables. Old is the constructed and sorted pairs list to be updated. }

procedure DIIPGPOL (g1,g2: LIST): LIST;
{Distributive integral polynomial g polynomial. g1 and g2 are integral polynomials in distributive representation. GPOL is the G-polynomial of g1 and g2. }

procedure DIIPSPOL2 (g1, g2, lcmHT, lcmHK: LIST): LIST;
{Distributive integral polynomial s polynomial. g1 and g2 are integral polynomials in distributive representation. lcmHT is the lcm of the highest terms of g1 and g2. lcmHK is the lcm of the highest coefficients of g1 and g2. polynomials in pair. SPol is the S-polynomial of g1 and g2. }

procedure DIIPLEXTAL (AL, g : LIST) : LIST;
{Distributive integral polynomial list extend array list. AL is an array list. g is a polynomial in distributive representation in r variables. Ag is the extended array list of AL. The list AL is modified. }

procedure DIIPLCPL4 (P : LIST; VAR CPL, AL : LIST);
{Distributive integral polynomial list construct pair list. P is a list of polynomials in distributive representation in r variables. CPL is the constructed pairs list, AL is the Array list. }

procedure DIIPALCMPC (AL, g1, g2, flag : LIST) : LIST;
{Distributive integral polynomial array list check and mark polynomials. AL is an array list. g1, g2 are polynomials in distributive representation in r variables. flag determines whether the pair will be marked as computed or only checked. 1 means to mark 0 only to check. The value 1 is returned if the pair (g1,g2) is already computed otherwise 0 is returned. }

procedure DIIPENF (P,varl,g: LIST): LIST;
{Distributive integral polynomial e-normal form. P is a list of non zero polynomials in distributive integral representation in r variables. g is a distributive integral polynomial. ENF is a polynomial such that g is e-reducible to ENF modulo P and ENF is in E-normalform modulo P. }

procedure DIIPREDDGB (P : LIST) : LIST;
{Distributive integral polynomial reduce D-groebner base. P is a list of non zero polynomials in distributive integral representation in r variables. REDDGB reduces the polynominials. It is nescessary that the highest monomials are pairwise disjoint. }

procedure DIIPSPOL (g1,g2: LIST): LIST;
{Distributive integral polynomial s polynomial. g1 and g2 are integral polynomials in distributive representation. SPol is the S-polynomial of g1 and g2. }

procedure DIIPDNF (P,varl,g: LIST): LIST;
{Distributive integral polynomial normal form. P is a list of non zero polynomials in distributive integral representation in r variables. g is a distributive integral polynomial. NF is a polynomial such that g is reducible to NF modulo P and NF is in D-normalform modulo P. }

procedure DIIPDGB (F, TF : LIST): LIST;
{Distributive integral polynomial D-groebner basis. F is a list of integral polynomials in distributive representation in r variables. G is the groebner basis of F. TF the trace flag. }

procedure DIIPEGB (F, TF : LIST): LIST;
{Distributive integral polynomial E-groebner basis. F is a list of integral polynomials in distributive representation in r variables. G is the groebner basis of F. TF the trace flag. }

9.6 DIP Integral Groebner Bases

procedure DIIGBA (PL,P,TF: LIST): LIST;
{Distributive integral polynomial groebner basis augmentation. P is a groebner basis of polynomials in distributive representation in r variables. p is a polynomial. PP is the groebner basis of (P,p). tf is the trace flag.}

procedure DIIGMI (P: LIST): LIST;
{Distributive minimal ordered groebner basis. P is a list of non zero integral polynomials in distributive representation in r variables. PP is the minimal normed and ordered groebner basis. }

procedure DIILIS (P: LIST): LIST;
{Distributive integral polynomial list irreducible set. P is a list of distributive integral polynomials, PP is the result of reducing each p element of P modulo P-(p) until no further reductions are possible. }

procedure DIIPGB (P,TF: LIST): LIST;
{Distributive integral polynomial groebner basis. P is a list of integral polynomials in distributive representation in r variables. PP is the groebner basis of P. tf is the trace flag.}

procedure DIIPNF (P,RPP,S: LIST): LIST;
{Distributive integral polynomial normal form. P is a list of non zero polynomials in distributive integral representation in r variables. S is a distributive integral polynomial. R is a polynomial such that S is reducible to R modulo P and R is in normalform with respect to p. }

procedure DIIPSP (A,B: LIST): LIST;
{Distributive integral polynomial s polynomial. A and B are integral polynomials in distributive representation. C is the S-polynomial of A and B. }

9.7 DIP Rational Function

procedure IFWRIT (R,V: LIST);
{Integral function write. R is an integral function. R is the variable list. R is written in the output stream. }

procedure RFDEN (R: LIST): LIST;
{Rational function denominator. R is a rational function. BL is the denominator of R, a positive integral polynomial in RL variables. }

procedure RFDIF (R,S: LIST): LIST;
{Rational function difference. R and S are rational functions. T=R-S. }

procedure RFEXP (A,NL: LIST): LIST;
{Rational function exponentiation. A is a rational function, n is a non-negative beta-integer. B=A**n. }

procedure RFFIP (RL,A: LIST): LIST;
{Rational function from integral polynomial. A is an integral polynomial in RL variables. R is the rational function A/1. }

procedure RFINV (R: LIST): LIST;
{Rational function inverse. R is a non-zero rational function. S=1/R. }

procedure RFNEG (R: LIST): LIST;
{Rational function negative. R is a rational function. S=-R. }

procedure RFNOV (R: LIST): LIST;
{Rational function number of variables. R is a rational function. RL is the number of variables of the numerator and denumerator of R. }

procedure RFNUM (R: LIST): LIST;
{Rational function numerator. R is a rational function. AL is the numerator of R, an integral polynomial. }

procedure RFONE (R: LIST): LIST;
{Rational function one. R is a rational function. s=1 if R=1, s=0 else. }

procedure RFPROD (R,S: LIST): LIST;
{Rational function product. R and S are rational functions. T=R*S. }

procedure RFQ (R,S: LIST): LIST;
{Rational function quotient. R and S are rational functions, S non-zero. T=R/S. }

procedure RFREAD (V: LIST): LIST;
{Rational function read. The rational function R is read from the input stream. V is the variable list. any preceding blanks are skipped. }

procedure RFRED (RL,A,B: LIST): LIST;
{Rational function reduction to lowest terms. A and B are integral polynomials in RL variables, B non-zero. R is the rational function A/B in canonical form. }

procedure RFSIGN (R: LIST): LIST;
{Rational function sign. R is a rational function. s=sign(R). }

procedure RFSUM (R,S: LIST): LIST;
{Rational function sum. R and S are rational functions. T=R+S. }

procedure RFWRIT (R,V: LIST);
{Rational function write. R is a rational function. V is the variable list. R is written in the output stream. }

9.8 DIP Rational Groebner Bases

procedure DIGBC3 (B,PLI,PLJ,EL: LIST): LIST;
{Distributive polynomial groebner basis criterion 3. B is a non empty list of reduction sets. pi and pj are distributive polynomials. e is the least common multiple of the leading exponent vectors of pi and pj. s=1 if the reduction of pi and pj is necessary s=0 else. }

procedure DIGBC4 (PLI,PLJ,EL: LIST): LIST;
{Distributive polynomial groebner basis criterion 4. pi and pj are polynomials in distributive representation. e is the least common multiple of the leading exponent vectors of pi and pj. s=1 if the reduction of pi and pj is necessary s=0 else. }

procedure DIGBMI (P: LIST): LIST;
{Distributive minimal ordered groebner basis. P is a list of non zero rational polynomials in distributive representation in r variables. PP is the minimal normed and ordered groebner basis. }

procedure DILCPL (P: LIST; VAR D,B: LIST);
{Distributive polynomial list construct pair list. P is list of polynomials in distributive representation in r variables. B is the polynomials pairs list and D is the pairs list. }

procedure DILUPL (PL,P,D,B: LIST): LIST;
{Distributive polynomial list update pair list. P is list of polynomials in distributive representation in r variables. B is the polynomials pairs list and D is the pairs list. p is a non zero polynomial in distributive representation. D, P and B are modified. DP is the updated pairs list. }

procedure DIRGBA (PL,P,TF: LIST): LIST;
{Distributive rational polynomial groebner basis augmentation. P is a groebner basis of polynomials in distributive representation in r variables. p is a polynomial. PP is the groebner basis of (P,p). t is the trace flag.}

procedure DIRGBR (P,TF: LIST): LIST;
{Distributive rational polynomial groebner basis recursion. P is a list of rational polynomials in distributive representation in r variables. PP is the groebner basis of P. t is the trace flag.}

procedure DIRLIS (P: LIST): LIST;
{Distributive rational polynomial list irreducible set. P is a list of distributive rational polynomials, PP is the result of reducing each p element of P modulo P-(p) until no further reductions are possible. }

procedure DIRPGB (P,TF: LIST): LIST;
{Distributive rational polynomials groebner basis. P is a list of rational polynomials in distributive representation in r variables. PP is the groebner basis of P. t is the trace flag.}

procedure DIRPNF (P,S: LIST): LIST;
{Distributive rational polynomial normal form. P is a list of non zero polynomials in distributive rational representation in r variables. S is a distributive rational polynomial. R is a polynomial such that S is reducible to R modulo P and R is in normalform with respect to P. }

procedure DIRPSP (A,B: LIST): LIST;
{Distributive rational polynomial S polynomial. A and B are rational polynomials in distributive representation. C is the S polynomial of A and B. }

procedure EVPLM (L1,L2: LIST): LIST;
{Exponent vector pair-list merge. L1 and L2 are pair-lists of exponent vectors in non decreasing order. L is the merge of L1 and L2. L1 and L2 are modified to produce L. }

procedure EVPLSO (A: LIST): LIST;
{Exponent vector pair-list sort. A is a list of pair-lists. B is the result of sorting A into non-decreasing order. Pairs of elements of A are merged. The list A is modified to produce B. }

9.9 DIP Ideal Real Root System

procedure DIGBSI (P,T,A: LIST): LIST;
{Distributive polynomial system algebraic number G basis sign. P is a goebner basis in inverse lexicographical term order in r variables (non empty), with all neccessary refinements. T=(t1,... ,ti) i le r, where tj=(vj,ij,pj) j=1, ...,i and v is the character list for the j-th variable, ij is an isolating interval for a real root of the univariate polynomial pjl. A is a distributive rational polynomial depending maximal on one variable. s is the sign of A as element of an algebraic extension of Q determined by P. }

procedure DIITNT (T: LIST): LIST;
{Distributive polynomial system interval tupel from norm tupel. T is a refined normalized tupel of a zero set with a final Goebner base of dimension 0. TP is a list of interval tupels for T. }

procedure DIITWR (TP,EPS: LIST);
{Distributive polynomial system interval tupels write. TP is a list of interval tupels of a zero set. EPS is LOG10 of the desired precision. }

procedure DINTWR (TP,EPS: LIST);
{Distributive polynomial system normalized tupels write. TP is a list of normalized tupels of a zero set. EPS is log10 of the desired precision. }

procedure DIROWR (V,P,EPS: LIST);
{Distributive polynomial system real root write. V is a variable list. P is a list (e,p). EPS is the desired precision. e is the multiplicity of the root, and p is an irreducible polynomial. }

procedure GBZSET (V,PP,EPS: LIST);
{Groebner base real zero set of zero dimensional ideal. V is a variable list. PP is a list of

distributive rational polynomials, PP is a Groebner base. EPS is is LOG10 of the desired precision. }

procedure RIRWRT (R,EPS: LIST);
{Rational interval refinement write. R=(v,i,p) where v is the variable character string, i is a rational interval containing only one real root of the polynomial p. EPS is the presicion epsilon. }

9.10 DIP Zero Dimensional Ideal

procedure DIRMPG (IL,F: LIST): LIST;
{Distributive rational minimal polynomial for a groebner basis. F is a groebner basis. PP is the minimal polynomial for the i-th variable for F. }

9.11 MAS Finite Field

procedure FFCOMP (R,S: LIST): LIST;
{Finite field comparison. R and S are finite field elements. t=0 if R=S, t=1 else. }

procedure FFDIF (p,M,AL,BL: LIST): LIST;
{Finite field difference. AL and BL are elements of F(p)(alpha) for some prime integer p and some algebraic number alpha. M is the minimal polynomial for alpha. CL=AL-BL.}

procedure FFEXP (p,M,A,NL: LIST): LIST;
{Finite field exponentiation. A is an element of F(p)(alpha) for some prime integer p and some algebraic number alpha. M is the minimal polynomial for alpha. nl is a non-negative beta-integer. B=A**nl.}

procedure FFFINT (p,M,A: LIST): LIST;
{Finite field element from integer. A is an integer. B is A converted to an element of F(p)(alpha), for some prime integer p and some algebraic number alpha. M is the minimal polynomial for alpha. }

procedure FFHOM (p,M,A: LIST): LIST;
{Finite field homomorpism. A is an univariate integral polynomial, B is A converted to an element of F(p)(alpha), for some prime integer p and some algebraic number alpha. M is the minimal polynomial for alpha. }

procedure FFINV (p,M,AL: LIST): LIST;
{Finite field inverse. AL is a nonzero element of F(p)(alpha) for some prime integer p and some algebraic number alpha. M is the minimal polynomial for alpha. BL=1/AL.}

procedure FFNEG (p,M,AL: LIST): LIST;
{Finite field negation. AL is an element of F(p)(alpha) for some prime integer p and some algebraic number alpha. M is the minimal polynomial for alpha. BL= -AL.}

procedure FFONE (R: LIST): LIST;
{Finite field one. R is a finite field element. s=1 if R=1, s=0 else. }

procedure FFPROD (p,M,AL,BL: LIST): LIST;
{Finite field product. AL and BL are elements of F(p)(alpha) for some prime integer p and some algebraic number alpha. M is the minimal polynomial of alpha. CL=AL+BL.}

procedure FFQ (p,M,AL,BL: LIST): LIST;
{Finite field quotient. AL and BL, BL nonzero, are elements of F(p)(alpha) for some prime integer p and some algebraic number alpha. M is the minimal polynomial for alpha. CL=AL/BL.}

procedure FFRAND (p,M,NL: LIST): LIST;
{Finite field element, random. n is a positive beta-integer. A random finite field element R of F(p)(alpha) for some prime integer p and some algebraic number alpha. M is the minimal polynomial for alpha. R is generated using IRAND(n). }

procedure FFREAD (V: LIST): LIST;
{Finite field read. The finite field element R is read from the input stream. V is the varaible list. Any preceding blanks are skipped. }

procedure FFSUM (p,M,AL,BL: LIST): LIST;
{Finite field sum. AL and BL are elements of F(p)(alpha) for some prime integer p and some algebraic number alpha. M is the minimal polynomial for alpha. CL=AL+BL.}

procedure FFWRITE (R, V: LIST);
{Finite field write. R is a finite field element. V is the varaible list. R is written to the output stream. }

9.12 MAS Polynomial GCD and RES System

procedure IPSFF (r,f: LIST): LIST;
{integral polynomial squarefree factorization f is a primitive polynomial in r variables, returns a list ((e1,p1),...,(ek,pk)), ei positive integers, pi squarefree polynomials, where f = p1**e1 * ... * pk**ek }

procedure IPFLMER (r,F1,F2: LIST): LIST;
{integral polynomial factorlist merge. F1 and F2 are factorlists ((e1,p1),...,(ek,pk)), ei positive integers, pi integer polynomials in r variables, as computed in IPSFF. returns the merge of F1 and F2. }

9.13 Universal Groebner Bases

procedure UGBBIN ();
{UGB input, execute and output. Diese hauptprozedur liest die eingabedatei ein (die polynome, die variablen und ihre anzahl durch pread, den parameter durch rdpar und die option durch execrd). Die funktion exeugb wird anschliessend aufgerufen. UGBBIN wird vom hauptprogramm main aufgerufen. }

procedure EXEUGB (L,I,V,PAR,OPT: LIST);
{UGB execute. Diese prozedur ruft, abhaengig von der option opt die prozeduren lf, plf, ugb und pugb. Diese prozeduren realisieren die verschiedenen optionen, die im abschnitt benutzerschnittstelle besprochen wurden. Die prozedur wird von der hauptprozedur ugbbin aufgerufen. }

procedure LF (L,I,V,PAR,OPT: LIST);
{UGB linear form. Die prozedur berechnet die linearformen nach der option LF. Vergleiche benutzerschnittstelle und 5.1.6. Die prozedur wird von exeugb aufgerufen. }

procedure PLF (L,I,V,PAR,OPT: LIST);
{UGB linear form with precomputed linear forms. Die prozedur berechnet die linearformen nach der option plf. Vergleiche benutzerschnittstelle und 5.1.7. Die prozedur wird von exeugb aufgerufen. }

procedure UGB (L,I,V,PAR,OPT: LIST);
{Universal Groebner base. Die prozedur berechnet eine universelle groebner basis nach der option ugb. Vergleiche benutzerschnittstelle und 5.2.4. Die prozedur wird von exeugb aufgerufen. }

procedure PUGB (L,I,V,PAR,OPT: LIST);
{Universal Groebner base with precomputed linear forms. Die prozedur berechnet eine universelle groebner basis nach der option pugb. Vergleiche benutzerschnittstelle und 5.2.5. Die prozedur wird von exeugb aufgerufen. }

procedure SUNIT1 (I: LIST);
{UGB set input unit 1. Diese prozedur stellt bei der option plf die richtige datei zum einlesen von linearformen bereit. I ist die anzahl der variablen. Die vorberechneten lineraformen sind je nach der anzahl der variablen in verschiedenen dateien gespeichert. Diese prozedur wird von der prozedur plf aufgerufen. }

procedure SUNIT2 (I: LIST);
{UGB set input unit 2. Diese prozedur stellt bei der option pugb die richtige datei zum einlesen von linearformen bereit. I ist die anzahl der variablen. Die vorberechneten lineraformen sind je nach der anzahl der variablen in verschiedenen dateien gespeichert. Diese prozedur wird von der prozedur pugb aufgerufen. }

procedure PREAD (VAR L,I,V: LIST);
{UGB polynomial read. Die polynome werden von der eingabedatei eingelesen. Diese funktion wird von der hauptprozedur ugbbin aufgerufen. }

procedure OPREAD (VAR PAR,OPT: LIST);
{UGB options and parameter read. Diese prozedur liest von der eingabedatei den parameter par (zustaendig fuer zwischenausgaben) durch die prozedur rdpar und die option opt (steht fuer lf, plf, ugb, pugb) durch die prozedur execrd. Diese prozedur wird von der hauptprozedur ugbbin aufgerufen. }

procedure EXPTU (L: LIST): LIST;
{UGB extract exponent vector list from polynomial list. Aus den polynomen wird die liste der terme berechnet. Da jeder term mit dem tupel seiner exponenten identifizert werden kann, wird die liste der exponententupel ausgegeben. Diese funktion wird von den prozeduren mklist, newdif, isneu, ug, pug, lf, plf, ugb und pugb. }

procedure MAKERN (Q: LIST): LIST;
{UGB rational exponent vector list from integer ev list. Makern transformiert die ganzzahlige struktur der exponententupel in eine rationalzahlige struktur diese funktion wird von den funktionen mklist, neulf und newdif aufgerufen. }

procedure SCMULT (I,U: LIST): LIST;
{UGB rational exponent vector rational number product. Hilfsfunktion zur berechnung vom skalarprodukt zwischen rationalzahligen vektoren. Diese funktion wird von der funktion mkset aufgerufen. }

procedure PDIF (R,S,DIFALT: LIST): LIST;
{UGB rational exponent vector list difference list, incremental. Berechnet (r-s) vereinigt mit (s-s).Diese prozedur ist im hinblick auf die berechnung von universellen groebner basen geschrieben. S entspricht der menge der neuen terme, die nach der reduktion entstehen. R enspricht der alten menge von termen. Da r-r in einem vorherigen schritt schon berechnet wurde, berechnet die prozedur nur r-r vereinigt mit s-s. Diese funktion wird von den funktionen projec, proj und newdif aufgerufen. }

procedure MKSET (R: LIST; VAR P2,P3,RR: LIST);
{UGB rational exponent vector list difference list. Berechnet die mengen p1, p2, p3 und die reduzierte menge von p - p wie sie im theoretischen teil der diplomarbeit beschrieben ist. Die eingabe ist r. P1 und p2 sind die projektionen. P3 ist die vereinigung von p1 und p2. Rr entspricht der reduzierten menge von p - p. Nur p2, p3 und rr werden zurueckgegeben. Diese funktion wird von den funktionen projec und proj aufgerufen. }

procedure PROJEC (R,S,DIFALT,OLDL,I,PAR: LIST): LIST;
{UGB projection to dimension 1. Berechnet die projektionen der eingabemenge r bis zur dimen-

sion 1. Die ausgabe ist ein stapel der tupel der form (p2,p) verschiedener dimensionen enthaelt. P2 wird in jeder dimension zur berechnung der schnitte und p zum vergleich von ordnungen benutzt. Diese prozedur ruft die prozeduren pdif und mkset auf. Die prozedur degre bestimmt fuer jede projektion den maximalen totalgrad der terme. Diese funktion wird von den funktionen lf, plf, ugb und newdif aufgerufen. }

procedure PROJ (R,S,DIFALT,OLDL,I: LIST; VAR D,B,DEG: LIST);
{UGB projection, one dimension. Berechnet die projektion der eingabemenge r auf eine niedrigere dimension. Die ausgabe besteht aus b (entspricht p2 in definition 2.2.1), d (entspricht der reduzierten differenz r - r), sowie deg. Diese prozedur ruft die prozeduren pdif und mkset auf. Die prozedur degre bestimmt den maximalen totalgrad deg der terme der projektion. Diese funktion wird von der funktion plf aufgerufen. }

procedure DIFF (R: LIST): LIST;
{UGB difference set for rational exponent vector list. Diese prozedur berechnet fuer die eingabemenge von tupel r die reduzierte menge r - r. das ergebnis wird in r zurueckgegeben. diese funktion wird von den funktionen mklist und pdif aufgerufen. }

procedure DIFF1 (R,S: LIST): LIST;
{UGB difference set for two rational exponent vector list. Berechnet fuer die eingabemengen r und s von exponeneten tupel die reduzierte menge r - s. das ergebnis wird in erg zurueckgegeben. diese funktion wird von der funktion pdif aufgerufen. }

procedure RNVDIF (A,B: LIST): LIST;
{UGB rational exponent vector difference. Berechnet die komponentenweise differenz von zwei rationalzahligen vektoren. die differenz von zwei rationalzahlen wird durch die prozedur rndif berechnet. diese funktion wird von den funktionen diff und diff1 aufgerufen. }

procedure SCPROD (A,B: LIST): LIST;
{UGB rational exponent vector scalar product. Berechnet das skalarprodukt von zwei rationalzahligen vektoren. die prozedur rnprod berechnet das produkt zweier rationalzahlen. die prozedur rnsum berechnet die summe von zwei rationalzahlen. das ergebnis wird in c zurueckgegeben. diese funktion wird von den funktionen cq2, cp2, pkegel und cspur aufgerufen. }

procedure SKPRO2 (A,B: LIST): LIST;
{UGB rational exponent vector scalar product with integer ev. Diese funktion ist eine spezielle skalarprodukt-funktion. da die ausgerechneten linearformen rationahlzahlig sind und die exponententupel ganzzahlig sind, werden diese zunaechst als rationahlzahlen dargestellt und dann das skalarprodukt gebildet. diese funktion wird von evlfcp aufgerufen. }

procedure LRNBMS (L: LIST): LIST;
{List of rational numbers bubble-merge sort. L is an arbitrary list of rational numbers, possibly with repetitions. M is the result of sorting L into non-decreasing order. A combination of bubble-sort and merge- sort is used. The list L is modified to produce M.}

procedure LRNBS (L: LIST);
{List of rational numbers bubble sort. L is an arbitrary list of rational numbers, with possible repetitions. L is sorted into non-decreasing order by the bubble-sort method. The list L, though not its location, is modified.}

procedure LRNM (L1,L2: LIST): LIST;
{List of rational numbers merge. L1 and L2 are arbitrary lists of rational numbers in non-decreasing order. L is the merge of L1 and L2. L1 and L2 are modified to produce L.}

procedure COMPLF (C,D,KLIST,NP,JP,M: LIST; VAR LFORM,KLISTP,J: LIST);
{UGB compute linear form from difference set. Diese prozedur berechnet fuer die linearform c und die menge der schnitte d die neuen linearformen. klist enthaelt die spuren der schon berechneten linearformen np entspricht der reduzierten menge p - p und wird dazu verwendet um die neuen spuren zu berechnen. lform enthaelt als ausgabe alle bisher berechneten linearformen. klistp die dazugehoerigen spuren. alle spuren sind verschieden. die funktion wird von der funktion mklf1

aufgerufen. }

procedure CQ2 (C,Q2,M: LIST): LIST;
{UGB linear form product with rational exponent vector list. Diese prozedur berechnet fuer eine linearform c und eine liste q2, beide der gleichen dimension, das produkt c * q2. die elemente von c * q2 bestehen aus dem skalarprodukt von c mit den einzelnen elementen von q2. da diese menge dazu verwendet wird, um die schnitte zu bilden, werden nur die negativen elemente gespeichert. die funktion wird von der funktion mklf1 aufgerufen. }

procedure RNVABS (A: LIST): LIST;
{Rational number list absolute values. Diese prozedur berechnet fuer die liste a von rationalzahlen den absolutbetrag ihrer komponenten. die prozedur rnabs berechnet den absolutbetrag einer rationalzahl. das ergebnis wird in der liste b zurueckgegeben. die funktion wird von den funktionen mklf1, mklf2 und mklf3 aufgerufen. }

procedure CUT (TR: LIST): LIST;
{UGB set of cuts. Berechnet fuer die eingabemenge tr die menge der schnitte d. fuer die inneren punkte wird das algebraische mittel gebildet. fuer die aeusseren punkte wird 1 addiert beziehungsweise die zahl halbiert. die funktion wird von den funktionen mklf1, mklf2 und mklf3 aufgerufen. }

procedure ALLELN (STAKK,L,KALT,I,PAR: LIST; VAR LF,NURLF: LIST);
{UGB all linear forms from stack of projections. Diese funktion berechnet aus dem stapel der projektionen stakk alle linearformen nurlf. die prozedur mklf1 wird aufgerufen. diese funktion wird von der prozedur lf aufgerufen. }

procedure MKLF1 (LFP,Q2,NP,M: LIST): LIST;
{UGB make new linear forms 1. Diese prozedur berechnet fuer eine liste von linearformen lfp die neuen linearformen newlf. die menge q2 wird dazu verwendet, die schnitte zu berechnen. die menge np dient dazu, die ueberfluessigen linearformen zu eliminieren. diese funktion wird von der prozedur plf und alleln aufgerufen. }

procedure NULRNV (A: LIST): LIST;
{Rational number vector null test. Diese prozedur ueberprueft ob ein vektor a der nullvektor ist. i ist 1 falls a der nullvektor ist ansonsten 0. diese funktion wird von der funktion mkset aufgerufen. }

procedure PKEGEL (C,N,KALT: LIST): LIST;
{UGB trace for linear form. Diese funktion berechnet die spur k bezueglich der linear- form c und der menge n in kodierter form. die spur wird nach der methode der wortkodierung (abschnitt 5.1.4) gebildet. diese funktion wird von der funktion complf aufgerufen. }

procedure COMPA1 (K,KLIST: LIST): LIST;
{UGB trace member in trace list. Diese funktion stellt fest ob eine spur k in einer liste von spuren vorhanden ist. j ist gleich 1 falls k in klist liegt, ansonsten 0. diese funktion wird von den funktionen complf, clf2, clf3 aufgerufen. }

procedure COMPA2 (K,A: LIST): LIST;
{UGB trace compare. Diese funktion ueberprueft zwei spuren k und a auf gleichheit. u ist gleich 1 falls a und k gleich sind, ansonsten 0. diese funktion wird von den funktionen compa1, dfp, dipmc2, zulfo und isneu aufgerufen. }

procedure LASTEL (Y: LIST): LIST;
{Last element. X ist das letzte element der liste y. }

procedure EVLRNBSO (A: LIST);
{Rational exponent vector list bubble sort. a is a list of rational exponent vectors, a is sorted with respect to the termordering defined in EVORD by the bubble-sort method, two exponent vectors with equal exponents will lead to an error. the list a but not its location, is modified.}

procedure EVRNGL (U,V: LIST): LIST;
{Rational exponent vector inverse graded lexicographical compare. u=(ul1, ..., ulrl), v=(vl1, ..., vlrl) are rational exponent vectors. tl=0 if u eq v. tl=1 if u gt v. tl=-1 if u lt v. eq, gt, lt with respect to the inverse graded lexicographical ordering of the exponent vectors. rl is the length of u and v.}

procedure EVRNC (U,V: LIST): LIST;
{Rational exponent vector compare. u=(ul1, ...,ulrl), v=(vl1, ...,vlrl) are exponent vectors. rl is the length of u and v. tl=0 if u eq v. tl=1 if u gt v. tl=-1 if u lt v. eq, gt, lt with respect to the ordering of the exponent vectors specified in the global variable EVORD. lexicographical, inverse lexicographical, graded lexicograhpical, inverse graded lexicographical orderings are possible. }

procedure DEGRE (Q: LIST): LIST;
{UGB total degree of a list of rational exponent vectors. Q ist eine liste von rationalzahligen tupeln. der maximale totalgrad der in q vorkommt wird berechnet. diese prozedur wird von den funktionen projec und proj aufgerufen. }

procedure RDPAR (): LIST;
{UGB read parameter. Diese funktion liest aus der eingabedatei den parameter par. zulaessige werte sind y oder n. bei y werden zwischen berechnungen ausgegeben, bei n nicht. diese funktion wird von der prozedur opread aufgerufen. }

procedure EXECRD (): LIST;
{UGB execution options read. Diese funktion liest aus der eingabedatei die auszufuehrende option. moegliche optionen sind lf plf ugb und pugb. vor der option muss das wort exec stehen. die option ist mit einem punkt abzuschliessen. beispiel exec ugb. diese funktion wurde mit geringfuegigen aenderungen aus der aldes-bibliothek entnommen. diese funktion wird von der hauptprozedur ugbbin aufgerufen. }

procedure SEENR (AC: LIST; VAR NR: LIST);
{UGB number of option. Diese funktion ermittelt fuer eine option ac eine schluesselzahl nr. die funktion stammt bis auf einige aenderungen aus der aldes-bibliothek. diese funktion wird von der funktion execrd aufgerufen. }

procedure LFGET (DEG,LF: LIST): LIST;
{UGB get linear form from list of linear forms. Diese funktion holt aus der liste lf der gespeicherten linearformen, abhaengig vom grad deg, die benoetigten linearformen. diese funktion wird von den funktionen plf, pugb und pug aufgerufen. }

procedure MKLF2 (LFP,Q2,NP,M,L: LIST; VAR NEWLF,LISTLF: LIST);
{UGB make new linear forms 2. Diese funktion ist genau analog zu mklf1. die linear- formen werden im gegensatz zu onenlf auch mit der zahl 1 als letzte komponente der linearformen berechnet. diese funktion wird von der funktion lfall aufgerufen. }

procedure CLF2 (C,D,KLIST,NP,JP,M,L: LIST; VAR LFORM,KLISTP,J,RECLF: LIST);
{UGB compute linear form from difference set 2. Diese funktion funktionniert genauso wie complf, mit dem unterschied, dass das element 1 als letzte komponente der linearform gespeichert wird. diese funktion wird von der funktion mklf2 aufgerufen. }

procedure CP2 (C,Q2: LIST): LIST;
{UGB linear form product with rational exponent vector list 2. Diese funktion funktionniert genauso wie cq2, mit dem unterschied, dass das element 1 als letzte komponente der linearform gespeichert wird. diese funktion wird von den funktionen mklf2 und mklf3 aufgerufen. }

procedure CSPUR (C,N,KALT: LIST): LIST;
{UGB trace for linear form 2. Diese funktion funktionniert genauso wie pkegel, mit dem unterschied, dass das element 1 als letzte komponente der linearform gespeichert wird. diese funktion wird von den funktionen clf2, clf3 und zulfo aufgerufen. }

procedure MKNEWP (P,POL,PRS: LIST): LIST;
{UGB make new critical pairs. Diese funktion aktualisiert die menge der paare prs der poly-

nomliste p um die paare der form (pol,f) wobei f aus p und pol ein polynom ist. das ergebnis ist ppairs. das buchberger-kriterium ist implementiert. diese funktion wird von der funktion gs2 aufgerufen. }

procedure MKPAIR (PP: LIST; VAR PAIRS: LIST);
{UGB make critical pairs for polynomial list. Diese funktion berechnet aus der liste pp von polynomen die menge der paare pairs. das buchberger-kriterium ist implementiert. diese funktion wird von der funktion gs1 aufgerufen. }

procedure MKSP1 (X,L,PAIRS,I,V: LIST; VAR D,PAIRSP: LIST);
{UGB compute next non-zero reduced S-polynomial. Diese funktion bildet bezueglich der linearform x und der polynommenge l aus der liste von paaren pairs solange ein s-polynom (dirpsp) und fuehrt es zu normalform (dirrnf) bis das s-polynom d nicht null ist oder die liste der paare pairsp leer ist. diese funktion wird von den funktionen ug und pug aufgerufen. }

procedure GS1 (LF,V,PAR: LIST): LIST;
{UGB generate stack of sorted polynomials and critical pairs 1. Lf ist ein tupel der form (a,l,k,n). dabei ist a eine linear- form, l eine liste von polynomen, k die dazugehoerige spur und n die reduzierte differenz p - p der entsprechenden menge von exponententupel. diese prozedur ordnet die polynome nach den linearformen und berechnet die menge der dazugehoerigen paare b. die ausgabe stak besteht aus tupeln der form (a,l,k,n,b). diese funktion wird von den funktionen ug und pug aufgerufen. }

procedure MERGE (STALT,STNEU: LIST): LIST;
{UGB merge stacks. Diese funktion mischt die zwei stapel stalt und stneu zu einem stapel stak wie in 5.2.3 beschrieben ist. diese funktion wird von der funktion neulf aufgerufen. }

procedure WRUGF (X,V,PAR: LIST): LIST;
{Write universal Groebner family. Diese funktion gibt eine berechnete universelle groebnerfamilie auf dem ausgabegeraet aus. es wird jeweils eine linearform und die dazugehoerige polynommenge ausgegeben. diese funktion wird von den prozeduren ugb und pugb aufgerufen. }

procedure WRUGB (UL,V: LIST);
{Write universal Groebner base. Diese funktion gibt eine berechnete universelle groebnerbasis ul auf dem ausgabegeraet aus. diese prozedur wird von den prozeduren ugb und pugb aufgerufen. }

procedure POLCOP (L: LIST): LIST;
{Two level list copy. Diese funktion macht eine kopie p der polynomliste l. diese funktion wird von den funktionen gs1, gs2 und wrugf aufgerufen. }

procedure DFP (A,B: LIST): LIST;
{UGB distributive rational polynomial difference. Diese funktion bildet aus den beiden polynomen a und b die distributive differenz a - b. das ergebnis ist cp. diese funktion unterscheidet sich von der in der aldes bibliothek vorhandenen funktion. sie berechnet die differenz bezueglich der in der globalen variablen EVORD gesetzten ordnung. diese funktion wird von den funktionen dirrnf und dirpsp aufgerufen. }

procedure SFP (A,B: LIST): LIST;
{UGB distributive rational polynomial sum. Diese funktion bildet aus den beiden polynomen a und b die distributive summe a + b. das ergebnis ist cp. diese funktion unterscheidet sich von der in der aldes bibliothek vorhandenen funktion. sie berechnet die differenz bezueglich der in der globalen variablen EVORD gesetzten ordnung. diese funktion wird von den funktionen dirrnf aufgerufen. }

procedure EVLFCP (L,U,V: LIST): LIST;
{UGB exponent vector linear form compare. Diese funktion vergleicht die exponententupel u und v zweier terme bezueglich der linearform l. das ergebnis t ist gleich 1 falls u groesser als v ist, 0 falls sie gleich sind und -1 ansonsten. diese funktion wird von der funktion evcomp aufgerufen. }

procedure PCOMP (X,Y: LIST): LIST;
{UGB distributive polynomial composition. Diese funktion bildet aus den beiden polynomen x

und y ein polynom z, sodass das polynom x der ersten teil, und y der zweite teil ist. diese funktion wird von den funktionen dfp und dipmc2 aufgerufen. }

procedure EVCOMP (U,V: LIST): LIST;
{UGB exponent vector compare. Diese funktion vergleicht die exponententupel u und v zweier terme bezueglich der termordnung, die in der globalen variable EVORD gespeichert ist. das ergebnis tl ist gleich 1 falls u groesser als v ist, 0 falls sie gleich sind und -1 ansonsten. }

procedure DIPMC2 (A,C,P: LIST): LIST;
{UGB distributive polynomial composition 2. Diese funktion bildet aus dem koeffizient a, dem term c und dem polynom p ein neues polynom dp. diese funktion wird von der funktione dirrnf aufgerufen. }

procedure DIRRNF (P,S,X,V: LIST): LIST;
{UGB distributive polynomial normalform. Diese funktion berechnet die normalform r eines polynoms s mit rationalen koeffizienten bezueglich der liste von polynomen p und der ordnung, die von der linearform x induziert wird. diese funktion wird von der funktione mksp1 aufgerufen. }

procedure DIRPSP (A,B,X: LIST): LIST;
{UGB distributive polynomial S-polynomial. Diese funktion berechnet das s-polynom c der polynome a und b bezueglich der ordnung, die von der linearform x induziert wird. diese funktion wird von der funktion mksp1 aufgerufen. }

procedure UG (LF,I,V,STAP,P,NURLF,PAR: LIST): LIST;
{Universal Groebner base. Diese funktion berechnet eine universelle groebner- familie ugf. lf sind tupel der form (a,l,k,n), wobei l die eingabemenge von polynomen, a eine von allen dazugehoerigen linearformen, k die spur und n die reduzierte differenz der eingabemenge der exponententupel ist. die berechnung realisiert die option ugb. diese funktion wird von der funktion ugb aufgerufen. }

procedure PUG (LF,I,V,P,DEGP,NURLF,PAR,LFQ: LIST): LIST;
{Universal Groebner base using precomputation. Diese funktion berechnet eine universelle groebner- familie ugf. lf sind tupel der form (a,l,k,n), wobei l die eingabemenge von polynomen, a eine von allen dazugehoerigen linearformen, k die spur und n die reduzierte differenz der eingabemenge der exponententupel ist. die berechnung realisiert die option pugb. diese funktion wird von der funktion pugb aufgerufen. }

procedure NEWL (LFTEMP,LFNEU,LFEND: LIST): LIST;
{UGB update linear forms from new terms. Lfneu ist die menge der linearformen auf der neuen menge von termen. die funktion stellt fest welche von diesen linearformen die alten fortsetzen und aktualisiert das zwischenergebnis lftemp durch lfp. die funktion wird auch nur aufgerufen wenn neue linearformen enstanden sind. diese funktion wird von den funktionen ug und pug aufgerufen. }

procedure ZULFO (LFNEU,X,LL,LFEND: LIST; VAR LFP,LF: LIST);
{UGB find admissible extensions of linear forms. Diese funktion stellt fest, welche linearformen aus lfneu die linearform von x fortsetzen. diese linearformen mit den aktualisierten daten (spur, paare) ersetzen dann x in ll. das ergebnis ist lfp. diese funktion wird von der funktion newl aufgerufen. }

procedure NONEWL (LFTEMP: LIST): LIST;
{UGB update linear forms without new terms. Diese funktion wird aufgerufen wenn keine linearformen entstanden sind. sie aktualisiert lftemp durch lfp. diese funktion wird von den funktionen ug und pug aufgerufen. }

procedure ISNEUL (LFALT,LFNEU,PAR: LIST): LIST;
{UGB new linear form test. Lfalt ist die alte liste von linearformen, lfneu die neue. diese funktion stellt fest ob neue linearformen entstanden sind (u gleich 1) oder nicht (u gleich 0). diese funktion wird von den funktionen ug und pug aufgerufen. }

procedure NEULF (STAP,DSUM,LSUM,I,V,PAR: LIST; VAR LFNEU,NEUST: LIST);
{UGB compute new linear forms from new terms. Diese funktion berechnet, nachdem neue terme entstanden sind, die neuen linearformen. die berechnung basiert auf die bereits beschriebenen funktionen und prozeduren zur berechnung von linearformen. dsum ist die liste der neuen s-polynome, lsum die liste der alten polynome, stap der alte stapel der projektionen. das ergebnis ist die liste der neuen linearformen lfneu und der neue stapel von projektionen neust. diese funktion wird von der funktion ug aufgerufen. }

procedure ISNEU (DSUM,LSUM,PAR: LIST; VAR SEMA,DD: LIST);
{UGB new terms test. Diese funktion stellt fest, ob neue terme in dusm entstanden sind. dsum ist die liste der neuen s-polynome in normalform, lsum die alte liste von polynomen. die ausgabe sema ist gleich 1 falls neue terme entstanden sind. ansonsten 0. dd ist die liste der neuen terme. diese funktion wird von den funktionen ug und pug aufgerufen. }

procedure TCOMP (X,Y: LIST): LIST;
{UGB list constructive conc. ??CCONC?? X und y sind zwei listen. diese prozedur konkatiniert sie zu einer liste z. diese funktion wird von der funktion isneu aufgerufen. }

procedure NEWDIF (L,D,DIFALT: LIST): LIST;
{UGB exponent vector list difference from polynomials. Diese funktion liest durch die funktion exptu die exponententupel der terme von l und d und berechnet mit hilfe der funktion pdif die differenz. fuer die berechnung der neuen differenz wird das schon berechnete ergebnis genutzt. diese funktion wird von den funktionen zulfo und nonewl aufgerufen. }

procedure GS2 (LF,V,PAR: LIST): LIST;
{UGB generate stack of sorted polynomials and critical pairs 2. Diese funktion funktioniert aehnlich zu gs1. sie ist wegen der uebersichtlichkeit getrennt geschrieben. die funktion aktualisiert die zwischenergebnisse lf (tupel der form (a,l,k,n,b) wie die ausgabe von gs1), d.h sie ordnet die polynome neu und aktualisiert die mengen von paaren. diese funktion wird von den funktionen ug und pug aufgerufen. }

procedure ALLLF (STAKK,KALT,I: LIST): LIST;
{UGB all linear forms from stack of projections and print. Die funktion funktionniert genauso wie lfall. hier werden nur die linearformen berechnet und ausgegeben. diese funktion wird von der funktion neulf aufgerufen. }

procedure LFALL (STAKK,L,KALT,I: LIST; VAR LISTLF,NURLF: LIST);
{UGB all linear forms from stack of projections 1. Diese funktion funktionniert genauso wie alleln, mit dem unterschied, dass das element 1 als letzte komponente der linearform gespeichert wird. diese funktion wird von der funktion ugb aufgerufen. }

procedure MKLF3 (LFP,Q2,NP,M: LIST): LIST;
{UGB make new linear forms 3. Diese funktionniert genauso wie mklf2. hier werden nur die linearformen (newlf) berechnet und ausgegeben. diese funktion wird von der funktion alllf aufgerufen. }

procedure CLF3 (C,D,KLIST,NP,JP,M: LIST; VAR LFORM,KLISTP,J: LIST);
{UGB compute linear form from difference set 3. Diese funktionniert genauso wie clf2. hier werden nur die linearformen (lform) berechnet und ausgegeben. diese funktion wird von der funktion mklf3 aufgerufen. }

procedure DO1 (LFP: LIST): LIST;
{UGB add last component to exponent vector. Um speicherplatz zu sparen wurden die linearformen ohne das 1 element als letzte komponente gespeichert. diese funktion fuegt fuer die liste der linearformen lfp das element 1 ein. das ergebnis ist lf1. diese funktion wird von der funktion pugb aufgerufen. }

procedure MKLIST (LF,L: LIST; VAR LFLIST,NURLF: LIST);
{UGB make trace and cuts. Diese funktion wird aufgerufen bei der option pugb. die liste der eingelesenen linearformen lf ist groesser als noetig. diese funktion reduziert diese linearformen,

sodass bezueglich der menge p von polynomen verschiedene ordnungen ergeben. nurlf ist dann das ergebnis. listlf besteht aus tupel der form (a,l,k,n) wie das ergebnis von lfall. diese funktion wird von der funktion pugb aufgerufen. }

procedure LDEG (L: LIST): LIST;
{Distributive polynomial list total degree. Diese funktion bestimmt fuer eine liste von polynomen l den maximalen totalgrad, der darin auftaucht. diese funktion wird von den funktionen pug und pugb aufgerufen. }

Chapter 10

SAC Ring Algorithms

10.1 SAC Algebraic Number Field

procedure AFDIF (AL,BL: LIST): LIST;
{Algebraic number field element difference. AL and BL are elements of Q(alpha) for some algebraic number alpha. CL=AL-BL.}

procedure AFINV (M,AL: LIST): LIST;
{Algebraic number field inverse. AL is a nonzero element of Q(alpha) for some algebraic number alpha. M is the rational minimal polynomial for alpha. BL=1/AL.}

procedure AFNEG (AL: LIST): LIST;
{Algebraic number field element negation. AL is an element of Q(alpha) for some algebraic number alpha. BL= -AL.}

procedure AFPROD (M,AL,BL: LIST): LIST;
{Algebraic number field element product. AL and BL are elements of Q(alpha) for some algebraic number alpha. M is the minimal polynomial of alpha. CL=AL+BL.}

procedure AFQ (M,AL,BL: LIST): LIST;
{Algebraic number field quotient. AL and BL are elements of Q(alpha) for some algebraic number alpha with BL nonzero. M is the minimal polynomial for alpha. CL=AL/BL.}

procedure AFSIGN (M,I,AL: LIST): LIST;
{Algebraic number field sign. M is the integral minimal polynomial of a real algebraic number alpha. I is an acceptable isolating interval for alpha. AL is an element of Q(alpha). SL=SIGN(AL).}

procedure AFSUM (AL,BL: LIST): LIST;
{Algebraic number field element sum. AL and BL are elements of Q(alpha) for some algebraic number alpha. CL=AL+BL.}

procedure RUPMRN (R: LIST): LIST;
{Rational univariate polynomial minimal polynomial of a rational number. R is a rational number. M is the rational minimal polynomial of R.}

10.2 SAC Extensions 1

procedure LCONC (L: LIST): LIST;
{List concatenation. L is a list (L sub 1 , ..., L sub n), n ge 0, such that each L sub i is a list. M eq CONC(L sub 1 , ..., L sub n). The lists L sub 1 , ..., L sub n are modified.}

procedure LEQUAL (A,B: LIST): LIST;
{List equality. A eq (A sub 1 , ..., A sub m), m ge 0, and B eq (B sub 1 , ..., B sub n), n ge 0, are two lists. b eq 1 if for each a sub i there is at least one B sub j such that A sub i eq B sub j, and for each B sub j there is at least one A sub i with B sub j eq a sub i. otherwise b eq 0.}

procedure LMERGE (A,B: LIST): LIST;
{List merge. A and B are lists of objects. C is the result of merging A and B.}

10.3 SAC Extensions 2

procedure RNBCR (A,B: LIST; VAR M,N,KL: LIST);
{Rational number binary common representation. A and B are binary rational numbers. If both A eq 0 and B eq 0, then M eq N eq K eq 0. If A eq 0, B ne 0, then M eq 0 and N and K are the unique integers such that B eq N cdot 2 sup k with N odd. If B eq 0, A ne 0, then N eq 0 and M and K are the unique integers such that A eq M cdot 2 sup K with M odd. If A ne 0 and B ne 0, then M,N, and K are the unique integers such that A eq M cdot 2 sup K and B eq N cdot 2 sup K with at least one of M and N odd.}

10.4 SAC Extensions 3

procedure CPLEXN (L: LIST; VAR I,M: LIST);
{Cartesian product, lexicographically next. L eq (L sub 1 , L sub 2 , ..., L sub 2n), n ge 1, is a list such that L sub 2i is a non-null list, and L sub 2i-1 is a non-null reductum of L sub 2i, for 1 le i le n. I is the element (first(L sub 1), first(L sub 3) , ..., first(L sub 2n-1)) of the cartesian product of L sub 2 , L sub 4 , ..., L sub 2n. If I is not the last element (in the inverse lexicographic ordering) of this cartesian product, then M is a list (M sub 1 , M sub 2 , ..., M sub 2n), with M sub 2i eq L sub 2i, M sub 2i-1 a non-null reductum of M sub 2i, for 1 le i le n, and (first(M sub 1), first(M sub 3) , ..., first(M sub 2n-1)) the lexicographically next element. If I is the last element, then M eq (). the list L is modified.}

procedure PERMCY (P: LIST): LIST;
{Permutation, cyclic. P is a list (P sub 1 , P sub 2 , ..., P sub n), n ge 0. PP eq (P sub 2 , P sub 3 , ..., P sub n , P sub 1).}

10.5 SAC Extensions 4

procedure IPINT (RL,A,BL: LIST): LIST;
{Integral polynomial integration. A is a non-zero integral polynomial in r variables, r ge 1, such that the integral of a with respect to its main variable is an integral polynomial. b is an integral polynomial in r-1 variables. B eq B(x sub 1 , ..., x sub r) is the integral of a with respect to its main variable, such that B(x sub 1 , ..., x sub r-1 ,0) eq b.}

procedure IUPIHT (A,NL: LIST): LIST;
{Integral univariate polynomial integer homothetic transformation. A is a non-zero univariate integral polynomial. n is a non-zero integer. B(x) is the primitive part of A(nx).}

procedure PCONST (RL,A: LIST): LIST;
{Polynomial constant. A(x sub 1 , ..., x sub r) is a polynomial in r variables, r ge 1. b eq 1 if a is a constant polynomial, otherwise b eq 0.}

procedure PSDSV (RL,A,IL,NL: LIST): LIST;
{Polynomial special decomposition, specified variable. A is a polynomial in r variables. 1 le i le

r and n is a beta-integer such that each exponent of x sub i occurring in a is divisible by n. B is A with each exponent e of x sub i replaced by e/n.}

procedure PUNT (RL,A: LIST): LIST;
{Polynomial univariate test. A eq A(x sub 1 , ..., x sub r) is a polynomial in r variables, r ge 1. b eq 2 if A has degree zero in all variables. b eq 1 if A has degree zero in x sub 2 , ..., x sub r, but positive degree in x sub 1. otherwise b eq 0.}

procedure RPDMV (RL,A: LIST): LIST;
{Rational polynomial derivative, main variable. A is a rational polynomial in r variables. B is the derivative of A with respect to its main variable.}

procedure RPMAIP (RL,A: LIST): LIST;
{Rational polynomial monic associate of integral polynomial. A is an integral polynomial in r variables, r ge 1. If A eq 0 then B eq 0. if A ne 0, let the integer a be the leading base coefficient of A. Then B eq (1/a) A, a monic rational polynomial.}

10.6 SAC Extensions 5

procedure IPCSFB (RL,A: LIST): LIST;
{Integral polynomial coarsest squarefree basis. A eq (A sub 1 , ..., A sub n), n ge 0, is a list of positive primitive integral polynomials in r variables, r ge 1, each of which is of positive degree in its main variable. B is a coarsest squarefree basis for A.}

procedure IPDSCR (RL,A: LIST): LIST;
{Integral polynomial discriminant. A is an integral polynomial in r variables, r ge 1, of degree greater than or equal to two in its main variable. B is the discriminant of A.}

procedure IPLCPP (RL,A: LIST; VAR C,P: LIST);
{Integral polynomial list of contents and primitive parts. A eq (A sub 1 , ..., A sub n), n ge 0, is a list of integral polynomials in r variables, r ge 1. C eq (C sub 1 , ..., C sub s), 0 le s le n, is a list such that for 1 le i le n, content(a sub i) eq c sub j for some j, 1 le j le s, if and only if content(a sub i) has positive degree in some variable. P eq (P sub 1 , ..., P sub m), 0 le m le n, is a list such that for 1 le i le n, PP(A sub i) eq P sub j for some j, 1 le j le m, if and only if PP(a sub i) has positive degree in its main variable.}

procedure IPPSC (RL,A,B: LIST): LIST;
{Integral polynomial principal subresultant coefficients. A and B are integral polynomials in r variables, r ge 1, of positive degree in the main variable. P is a list of the principal subresultant coefficients of the second kind of A and B.}

procedure IPSFBA (RL,A,B: LIST): LIST;
{Integral polynomial squarefree basis augmentation. A is a primitive positive squarefree integral polynomial in r variables, r ge 1, of positive degree in its main variable. B eq (B sub 1 , ..., B sub s), s ge 0, is a squarefree integral polynomial basis in r variables. BS is a coarsest squarefree basis for (A,B sub 1 , ..., B sub s).}

procedure ISPSFB (RL,A: LIST): LIST;
{Integral squarefree polynomial squarefree basis. A eq (A sub 1 , ..., A sub n), n ge 0, is a list of positive primitive squarefree integral polynomials in r variables,r ge 1, each of which is of positive degree in its main variable. B is a coarsest squarefree basis for A.}

procedure IUPRC (A,B: LIST; VAR C,R: LIST);
{Integral univariate polynomial resultant and cofactor. A and B are univariate integral polynomials of positive degree. R is the resultant of A and B. C is a univariate integral polynomial such that for some univariate integral polynomial D, AD+BC eq R.}

procedure MUPRC (PL,A,B: LIST; VAR C,RL: LIST);
{Modular univariate polynomial resultant and cofactor. p is a prime beta-digit. A and B are

univariate polynomials over Z sub p of positive degree. R is the resultant of A and B, an element of Z sub p. C is a univariate polynomial over Z sub p such that for some univariate polynomial D over Z sub p, AD+BC eq R.}

10.7 SAC Extensions 6

procedure IPFSFB (RL,A: LIST): LIST;
{Integral polynomial finest squarefree basis. A eq (A sub 1 , ..., A sub n), n ge 0, is a list of positive primitive integral polynomials in r variables, r ge 1, each of which is of positive degree in its main variable. B is a finest squarefree basis for A.}

10.8 SAC Extensions 7

procedure IPRICL (A: LIST): LIST;
{Integral polynomial real root isolation, Collins-Loos algorithm. A is an integral polynomial. L is a strong isolation list for A.}

procedure IPRRII (A,AP,DL,LP: LIST): LIST;
{Integral polynomial real root isolation induction. A is a primitive positive univariate integral polynomial of positive degree. AP is the derivative of A. D is a binary rational real root bound for A. LP is a strong isolation list for AP. L is a strong isolation list for A.}

procedure IPRRRI (A,B,I,SL1,TL1: LIST): LIST;
{Integral polynomial relative real root isolation. A and B are univariate integral polynomials. I is a left-open, right-closed interval (a sub 1 ,a sub 2) where al sub 1 and al sub 2 are binary rational numbers with al sub 1 lt al sub 2. A and B have unique roots, alpha and beta respectively, in I, each of odd multiplicity and with alpha ne beta. sl sub 1 eq sign(A(al sub 1 +)) and tl sub 1 eq sign(B(al sub 1 +)). is eq (al sub 1 sup * ,al sub 2 sup *) is al left-open, right-closed subinterval of I with al sub 1 sup * and al sub 2 sup * binary rational numbers and al sub 1 sup * lt al sub 2 sup *, such that is contains alpha but not beta.}

procedure IPSIFI (A,I: LIST): LIST;
{Integral polynomial standard isolating interval from isolating interval. I is an interval with binary rational endpoints, which is either left-open and right-closed or a one-point interval. A is a univariate integral polynomial which has a unique root alpha of odd multiplicity in I. If I is a one-point interval, then J=I. If I is left-open and right-closed, then J is either a standard left-open and right-closed subinterval of I containing alpha, or if alpha is a binary rational number, J may possibly instead be the one-point interval (alpha , alpha).}

procedure ISFPIR (A,I,KL: LIST): LIST;
{Integral squarefree polynomial isolating interval refinement. A is a squarefree univariate integral polynomial. I is an isolating interval for a real root alpha of A. k is a nonnegative beta -integer. J is a subinterval of I isolating alpha with length less than 10 sup -k.}

procedure IUPVOI (A,I: LIST): LIST;
{Integral univariate polynomial, variations for open interval. A is a non-zero integral univariate polynomial. I is an open interval (a,b) with a and b binary rational numbers such that a lt b. Let t(z) be the transformation mapping the right half-plane onto the circle having I as diameter. Let B(X) eq A(t(X)). then v is the number of sign variations in the coefficients of B.}

10.9 SAC Extensions 8

procedure AFCOMP (MB,I,AL,BL: LIST): LIST;
{Algebraic number field comparison. MB is the integral minimal polynomial of a real algebraic number alpha. I is an acceptable isolating interval for alpha. a and b are elements of Q(alpha). t eq SIGN(a-b).}

procedure AFFINT (M: LIST): LIST;
{Algebraic number field element from integer. M is an integer. A is M represented as an element of an algebraic number field.}

procedure AFFRN (R: LIST): LIST;
{Algebraic number field element from rational number. R is a rational number. A is R represented as an element of an algebraic number field.}

procedure AFPAFP (RL,M,AL,B: LIST): LIST;
{Algebraic number field polynomial algebraic number field element product. M is the rational minimal polynomial of an algebraic number alpha. a is an element of Q(alpha). B is a polynomial over Q(alpha) in r variables, r ge 1. C eq a cdot B.}

procedure AFPAFQ (RL,M,A,BL: LIST): LIST;
{Algebraic number field polynomial algebraic number field element quotient. M is the rational minimal polynomial of an algebraic number alpha. A is a polynomial over Q(alpha) in r variables, r ge 1. b is an element of Q(alpha). C eq A/b.}

procedure AFPBRI (M,MB,I,L: LIST): LIST;
{Algebraic number field polynomial basis real root isolation. M is the rational minimal polynomial of a real algebraic number alpha. MB is the integral minimal polynomial of alpha. I is an acceptable isolating interval for alpha. L is a nonempty squarefree basis (A sub 1 ,... , A sub n) of univariate polynomials over Q(alpha). N is a list (i sub 1 ,b sub 1 ,... , i sub m ,b sub m), m ge 0, where i sub 1 lt i sub 2 lt ... lt i sub m are strongly disjoint isolating intervals for all the real roots of a eq prod from (j eq 1) to n (a sub j). each i sub i has binary rational endpoints and is left open and right closed. b sub i is the unique a sub j which has a root in i sub i.}

procedure AFPCLL (M,MB,I,A: LIST): LIST;
{Algebraic number field polynomial real root isolation, Collins-Loos algorithm, list output version. M is the rational minimal polynomial of a real algebraic number alpha. MB is the integral minimal polynomial of alpha. I is an acceptable isolating interval for alpha. A is a monic univariate polynomial of degree n ge 0 over Q(alpha). If n eq 0 then L eq (). If n gt 0, then L eq (L sub 0 ,... , L sub n-1), where L sub i is a strong isolation list for the real roots of der sup i (A).}

procedure AFPDIF (RL,A,B: LIST): LIST;
{Algebraic number field polynomial difference. A and B are polynomials in r variables, r ge 0, over Q(alpha), for some algebraic number alpha. C=A-B.}

procedure AFPDMV (RL,M,A: LIST): LIST;
{Algebraic number field polynomial derivative, main variable. M is the rational minimal polynomial of an algebraic number alpha. A is a polynomial over Q(alpha) in r variables, r ge 1. B is the derivative of a with respect to its main variable.}

procedure AFPEMV (RL,M,A,AL: LIST): LIST;
{Algebraic number field polynomial evaluation of main variable. M is the rational minimal polynomial of an algebraic number alpha. A is a polynomial over Q(alpha) in r variables, r ge 1. a is an element of Q(alpha). B(x sub 1 ,... , x sub r-1) eq A(x sub 1 ,... , x sub r-1 ,a).}

procedure AFPEV (RL,M,A,IL,AL: LIST): LIST;
{Algebraic number field polynomial evaluation. M is the rational minimal polynomial of an algebraic number alpha. A is a polynomial in r ge 1 variables over Q(alpha). i satisfies 1 le i le r, and a is an element of Q(alpha). B(x sub 1 ,... , x sub i-1 , x sub i+1 ,... , x sub r) eq A(x sub 1 ,... , x sub i-1 , a , x sub i+1 ,... , x sub r).}

procedure AFPFIP (RL,A: LIST): LIST;
{Algebraic number field polynomial from integral polynomial. A is an integral polynomial in r variables, r ge 1. B is a represented as a polynomial over an algebraic number field.}

procedure AFPFRP (RL,A: LIST): LIST;
{Algebraic number field polynomial from rational polynomial. A is a rational polynomial in r variables, r ge 1. B is a represented as a polynomial over an algebraic number field.}

procedure AFPINT (RL,M,A,BL: LIST): LIST;
{Algebraic number field polynomial integration. M is the rational minimal polynomial of an algebraic number alpha. A is a nonzero polynomial over Q(alpha) in r variables, r ge 1. b is a polynomial over Q(alpha) in r-1 variables. B eq B(x sub 1 ,... , x sub r) is the integral of a with respect to its main variable, such that B(x sub 1 ,... , x sub r-1 ,0) eq b.}

procedure AFPME (RL,M,A,BL: LIST): LIST;
{Algebraic number field, polynomial multiple evaluation. M is the rational minimal polynomial of an algebraic number alpha. A eq A(x sub 1 ,... , x sub r) is a polynomial in r ge 1 variables over Q(alpha). b eq (b sub 1 ,... , b sub k) is a list of k elements of Q(alpha) for some k, 1 le k le r. B eq A(b sub 1 ,... , b sub k ,x sub k+1 ,... , x sub r), an element of Q(alpha)(x sub k+1 ,... , x sub r).}

procedure AFPMON (RL,M,A: LIST): LIST;
{Algebraic number field polynomial monic. A is a polynomial in r variables, r ge 1, over Q(alpha) for some algebraic number alpha. M is the rational minimal polynomial of alpha. If A is nonzero then AP is the monic polynomial over Q(alpha) similar to A. If A eq 0 then AP eq 0.}

procedure AFPMPR (M,MB,I,B,J,L: LIST; VAR JS,JL: LIST);
{Algebraic number field polynomial minimal polynomial of a real root. M is the rational minimal polynomial of a real algebraic number alpha. MB is the integral minimal polynomial of alpha. I is an acceptable isolating interval for alpha. J is an interval with binary rational number endpoints which is either left-open and right-closed, or a one-point interval. B is a univariate polynomial over Q(alpha) having a unique root beta of odd multiplicity in j. L is a nonempty list of positive irreducible univariate integral polynomials exactly one of which has beta as a root. j is the index in L of the unique element n of L having beta as a root, and js is a subinterval of j with binary rational endpoints which is an isolating interval for beta as a root of n. js is either left-open and right-closed or a one-point interval.}

procedure AFPNEG (RL,A: LIST): LIST;
{Algebraic number field polynomial negative. A is a polynomial in r variables, r ge 0, over Q(alpha) for some algebraic number alpha. B=-A.}

procedure AFPNIP (MB,A: LIST): LIST;
{Algebraic number field polynomial normalize to integral polynomial. MB is the integral minimal polynomial of an algebraic number alpha. A is a univariate polynomial over Q(alpha) of positive degree. l is a list (l sub 1 ,... , l sub n), n ge 1, of the positive irreducible factors of positive degree of a univariate integral polynomial which has among its roots the roots of A.}

procedure AFPPR (RL,M,A,B: LIST): LIST;
{Algebraic number field polynomial product. A and B are polynomials in r variables, r ge 0, over Q(alpha) for some algebraic number alpha. M is the rational minimal polynomial of alpha. C=A*B.}

procedure AFPQR (RL,M,A,B: LIST; VAR Q,R: LIST);
{Algebraic number field polynomial quotient and remainder. A and B, B ne 0, are polynomials in r variables, r ge 1, over Q(alpha), for some algebraic number alpha. M is the rational minimal polynomial of alpha. Q and R are the unique algebraic number field polynomials such that either B divides A, Q eq B/A, and r eq 0 or else B does not divide A and A eq BQ+R with degree(R) minimal.}

procedure AFPRCL (M,MB,I,A: LIST): LIST;
{Algebraic number field polynomial real root isolation, Collins-Loos algorithm. M is the rational minimal polynomial of a real algebraic number alpha. MB is the integral minimal polynomial of alpha. I is an acceptable isolating interval for alpha. A is a monic univariate polynomial of degree n ge 0 over Q(alpha). L is a strong isolation list for the real roots of a.}

procedure AFPRII (M,MB,J,A,AP,DL,LP: LIST): LIST;
{Algebraic number field polynomial real root isolation induction. M is the rational minimal polynomial of a real algebraic number alpha. MB is the integral minimal polynomial of alpha. J is an acceptable isolating interval for alpha. A is a positive univariate polynomial over Q(alpha) of positive degree. AP is the derivative of A. d is a binary rational real root bound for A. LP is a strong isolation list for AP. L is a strong isolation list for A.}

procedure AFPRLS (M,MB,I,A1,A2,L1,L2: LIST; VAR LS1,LS2: LIST);
{Algebraic number field polynomial real root list separation. M is the rational minimal polynomial of a real algebraic number alpha. MB is the integral minimal polynomial of alpha. I is an acceptable isolating interval for alpha. A1 and A2 are univariate polynomials over Q(alpha) with no common roots and real roots of only odd multiplicity. L1 and L2 are strong isolation lists for the real roots of A1 and A2 respectively. Let L1 eq (i sub 1,1 ,m sub 1,1 ,... , i sub (1,r sub 1),m sub (1,r sub 1)), L2 eq (i sub 2,1 ,m sub 2,1 ,... , i sub (2,r sub 2),m sub (2,r sub 2)). Then i sub 1,1 lt i sub 1,2 lt ... lt i sub (1,r sub 1) and i sub 2,1 lt i sub 2,2 lt ... lt i sub (2,r sub 2) . l sub 1 sup * eq (i sub 1,1 sup * ,m sub 1,1 ,... , i sub (1,r sub 1) sup * ,m sub (1,r sub 1)) and l sub 2 sup * eq (i sub 2,1 sup * ,m sub 2,1 ,... , i sub (2,r sub 2) sup * ,m sub (2,r sub 2)), where i sub i,j sup * is a binary rational subinterval of i sub i,j containing the root of a sub i in i sub i,j. each i sub 1,j sup * is strongly disjoint from each i sub 2,j sup *.}

procedure AFPRRI (M,MB,I,A,B,J,SL1,TL1: LIST): LIST;
{Algebraic number field polynomial relative real root isolation. M is the rational minimal polynomial of a real algebraic number alpha. MB is the integral minimal polynomial of alpha. I is an acceptable isolating interval for alpha. A and B are univariate polynomials over Q(alpha). J is a left open, right closed interval (a sub 1 ,a sub 2) where al sub 1 and al sub 2 are binary rational numbers with al sub 1 lt al sub 2. A and B have unique roots, alpha and beta respectively, in J, each of odd multiplicity and with alpha ne beta. sl sub 1 eq sign(a(al sub 1 +)) and tl sub 1 eq sign(b(al sub 1 +)). js eq (al sub 1 sup * ,al sub 2 sup *) is al left-open, right-closed subinterval of j with al sub 1 sup * and al sub 2 sup * binary rational numbers and al sub 1 sup * lt al sub 2 sup *, such that js contains alpha but not beta.}

procedure AFPRRS (M,MB,I,A1,A2,I1,I2: LIST; VAR IS1,IS2,SL: LIST);
{Algebraic number field polynomial real root separation. M is the rational minimal polynomial of a real algebraic number alpha. MB is the integral minimal polynomial of alpha. I is an acceptable isolating interval for alpha. A1 and A2 are univariate integral polynomials of positive degrees over Q(alpha). I1 and I2 are intervals with binary rational number endpoints, each of which is either left-open and right-closed, or a one-point interval. I1 contains a unique root alpha sub 1 of A1 of odd multiplicity, and I2 contains a unique root alpha sub 2 ne alpha sub 1 of A2 of odd multiplicity. I sub 1 sup * and I sub 2 sup * are binary rational subintervals of I1 and I2 containing alpha sub 1 and alpha sub 2 respectively, with I sub 1 sup * and I sub 2 sup * strongly disjoint. If I1 is left-open and right-closed then so is I sub 1 sup *, and similarly for I2 and I sub 2 sup *. s eq -1 if I sub 1 sup * lt I sub 2 sup *, and s eq 1 if I sub 1 sup * gt I sub 2 sup *.}

procedure AFPSUM (RL,A,B: LIST): LIST;
{Algebraic number field polynomial sum. A and B are polynomials over Q(alpha) in r variables, r ge 1, for some algebraic number alpha. C=A+B.}

procedure AFSUPB (M,A: LIST): LIST;
{Algebraic number field squarefree univariate polynomial squarefree basis. M is the rational minimal polynomial of an algebraic number alpha. A eq (a sub 1 ,... , a sub n), n ge 0, is a list of monic squarefree univariate polynomials over Q(alpha), each of which is of positive degree.

B is a coarsest squarefree basis for A.}

procedure AFUPBA (M,A,B: LIST): LIST;
{Algebraic number field univariate polynomial squarefree basis augmentation. M is the rational minimal polynomial of an algebraic number alpha. A is a monic squarefree univariate polynomial over Q(alpha), of positive degree. B eq (b sub 1 ,... , b sub s), s ge 0, is a squarefree basis of univariate polynomials over Q(alpha). BS is a coarsest squarefree basis for (a,b sub 1 ,... , b sub s).}

procedure AFUPCB (M,A: LIST): LIST;
{Algebraic number field univariate polynomial coarsest squarefree basis. M is the rational minimal polynomial of an algebraic number alpha. A eq (a sub 1 ,... , a sub n), n ge 0, is a list of monic univariate polynomials over Q(alpha), each of which is of positive degree. B is a coarsest squarefree basis for A.}

procedure AFUPGC (M,A,B: LIST; VAR C,AB,BB: LIST);
{Algebraic number field univariate polynomial greatest common divisor and cofactors. A and B are univariate polynomials over Q(alpha) for some algebraic number alpha. M is the rational minimal polynomial of alpha. C eq gcd(A,B), a monic polynomial. If C ne 0, then AB eq A/C and BB eq B/C, otherwise AB eq 0 and BB eq 0. }

procedure AFUPGS (M,A: LIST): LIST;
{Algebraic number field polynomial greatest squarefree divisor. M is the rational minimal polynomial of an algebraic number alpha. A is a univariate polynomial over Q(alpha). If A eq 0 then B eq 0. Otherwise B is the monic associate of the greatest squarefree divisor of A.}

procedure AFUPRB (MB,I,A: LIST): LIST;
{Algebraic number field univariate polynomial root bound. MB is the integral minimal polynomial of a real algebraic number alpha. I is an acceptable isolating interval for alpha. A is a monic univariate polynomial over Q(alpha) of positive degree. B is a binary rational number which is a root bound for A. If A(x) eq x sup n + sum from (i eq 0) to n-1 (a sub i x sup i), then B is the smallest power of 2 such that 2 cdot (abs(a sub n-k)) sup 1/k le B for 1 le k le n. If a sub n-k eq 0 for 1 le k le n then B eq 1.}

procedure AFUPRL (M,A: LIST): LIST;
{Algebraic number field polynomial, root of a linear polynomial. A is an element of Q(alpha)(x) of degree one, for some algebraic number alpha. M is the rational minimal polynomial of alpha. a is the unique element of Q(alpha) such that A(a) eq 0.}

procedure AFUPSF (M,A: LIST): LIST;
{Algebraic number field univariate polynomial squarefree factorization. M is the rational minimal polynomial of an algebraic number alpha. A is a monic univariate polynomial over Q(alpha) of positive degree. L is the list ((e sub 1 ,a sub 1) ,... , (e sub k ,a sub k)), where A eq prod from (i eq 1) to k (a sub i sup (e sub i)) is the squarefree factorization of A, with 1 le e sub 1 lt e sub 2 le ... lt e sub k and each a sub i a monic squarefree polynomial of positive degree.}

procedure AFUPSR (M,MB,I,A,CL: LIST): LIST;
{Algebraic number field univariate polynomial, sign at a rational point. M is the rational minimal polynomial of a real algebraic number alpha. MB is the integral minimal polynomial of alpha. I is an acceptable isolating interval for alpha. A is a univariate polynomial over Q(alpha). c is a rational number. s eq sign(A(c)).}

procedure ANDWR (M,I,NL: LIST);
{Algebraic number decimal write. M is the integral minimal polynomial of a real algebraic number alpha. I is an acceptable isolating interval for alpha. n is a nonnegative integer. alpha is approximated by a rational number r with inaccuracy of approximation at most 10 sup -(n+1). Then r is approximated by a decimal fraction d with n decimal digits following the decimal point and d is written in the output stream. The inaccuracy of the approximation of d to r is at most (1/2) 10 sup -n. If abs(d) gt abs(r) then the last digit is followed by

$= -.If abs(d) ltabs(r) then by = +.\}$

procedure ANFAF (M,I,AL: LIST; VAR N,J: LIST);
{Algebraic number from algebraic number field element. M is the integral minimal polynomial of a real algebraic number alpha. I is an acceptable isolating interval for alpha. a is an element of Q(alpha). N is the integral minimal polynomial of a, and J is an acceptable isolating interval for a.}

procedure ANIIPE (MB,I,NB,J,TL,L: LIST; VAR S,KL,K: LIST);
{Algebraic number isolating interval for a primitive element. MB is the integral minimal polynomial of a real algebraic number alpha. I is a binary rational isolating interval for alpha which is either left-open and right-closed or a one-point interval. NB is the integral minimal polynomial of a real algebraic number beta. J is a binary rational isolating interval for beta which is either left-open and right-closed or a one-point interval. t is an integer such that Q(alpha +t beta) eq Q(alpha , beta). If degree(MB) eq 1 and degree(NB) eq 1, then L is a list containing a primitive positive integral polynomial p of degree 1, s eq p, k eq 1, and k is a binary rational isolating interval for the real root of p which is either left-open and right-closed or a one-point interval. If degree(mb) eq 1, degree(NB) gt 1, then l eq (NB), S eq NB, k eq 1, and k eq j. If degree(MB) gt 1, degree(NB) eq 1, then L eq (MB), S eq MB, k eq 1, and k eq i. If degree(MB) gt 1, degree(NB) gt 1, then L is a nonempty list of positive irreducible univariate integral polynomials exactly one of which has alpha +t beta as a root. S is the element of L having alpha +t beta as a root, k is the index of S in L, and k is a left-open, right-closed binary rational isolating interval for alpha +t beta as a root of S.}

procedure ANPEDE (MB,NB: LIST; VAR TL,S,T: LIST);
{Algebraic number primitive element for a double extension. MB eq MB(x) is the integral minimal polynomial of a real algebraic number alpha. NB eq NB(x) is the integral minimal polynomial of a real algebraic number beta. t is an integer such that Q(alpha +t beta) eq Q(alpha , beta). If degree(MB) eq 1 and degree(NB) eq 1, then S eq (x), a list of length of length 1 containing the polynomial x. If degree(MB) eq 1 and degree(NB) gt 1, then S eq (NB). If degree(MB) gt 1 and degree(NB) eq 1, then S eq (MB). If degree(MB) gt 1 and degree(NB1) gt 1, then S is a list of the integral minimal polynomials of all algebraic numbers of the form alpha sub i + t beta sub j, where alpha sub i is some conjugate of alpha and beta sub j is some conjugate of beta. Where n ge 1 is the length of S, t is a list (m sub 1 sup * ,n sub 1 sup * ,... , m sub n sup * , n sub n sup *). For 1 le k le n, where gamma sub k is a root of s sub k, m sub k sup * is the representation of alpha as an element of Q(gamma sub k) and n sub k sup * is the representation of beta as an element of Q(gamma sub k).}

procedure ANREPE (M,MB,A,B: LIST): LIST;
{Algebraic number represent element of a primitive extension. M is the rational minimal polynomial of an algebraic number gamma. MB is the integral minimal polynomial of gamma. A and B are elements of Q(gamma) (y) which can be and are represented as bivariate integral polynomials, i.e. as elements of Z(x,y). A eq AP(x-ty) for a minimal polynomial AP of an algebraic number alpha, and B eq B(y) is the minimal polynomial of an algebraic number beta. gamma is a primitive element for alpha and beta. B is a univariate rational polynomial which is the representation for beta as an element of Q(gamma).}

procedure APDWR (M,I,BL,NL: LIST);
{Algebraic point, decimal write. M,I, and b constitute the representation of an algebraic point in r-dimensional euclidean space for some r ge 1. n is a nonnegative integer. For each coordinate b sub i of b, b sub i is represented by a rational number r sub i with inaccuracy of approximation at most 10 sup -(n+1). Then r sub i is approximated by a decimal fraction d sub i with n decimal digits following the decimal point, and d sub i is written in the output stream. The inaccuracy of the approximation of d sub i to r sub i is at most (1/2) 10 sup -n.}

procedure IPAFME (RL,M,A,BL: LIST): LIST;
{Integral polynomial, algebraic number field multiple evaluation. A is an integral polynomial in

r variables, r ge 1. M is the rational minimal polynomial of an algebraic number alpha. b eq (b sub 1 ,... , b sub k) is a list of k elements of Q(alpha), for some k, 1 le k le r. b eq a(b sub 1 ,... , b sub k , x sub k+1 ,... , x sub r), an element of Q(alpha)(x sub k+1 ,... , x sub r).}

procedure IUPMRN (R: LIST): LIST;
{Integral univariate polynomial minimal polynomial of a rational number. R is a rational number. M is the integral minimal polynomial of R.}

procedure RPAFME (RL,M,A,BL: LIST): LIST;
{Rational polynomial, algebraic number field multiple evaluation. A is a rational polynomial in r variables, r ge 1. M is the rational minimal polynomial of an algebraic number alpha. b eq (b sub 1 ,... , b sub k) is a list of k elements of Q(alpha), for some k, 1 le k le r. B eq A(b sub 1 ,... , b sub k , x sub k+1 ,... , x sub r), an element of Q(alpha)(x sub k+1 ,... , x sub r).}

10.10 SAC Modular Univariate Polynomial Factorization

procedure MCPMV (NL,L: LIST): LIST;
{Matrix of coefficients of polynomials, with respect to main variable. L is a list (L(1), ...,L(k)) of k, ge 1, non-zero polynomials with degrees less than n. n is a positive beta-integer. M is the matrix with m(1,il)+m(2,i)*x+ ...+m(n,i)*x**(n-1)=L(i) for 1 le i le k.}

procedure MIUPSE (M,A,B,S,T,C: LIST; VAR U,V: LIST);
{Modular integral univariate polynomial, solution of equation. M is a positive integer. A,B,S,T and C belong to Z sub M (x) with ldcf(A) a unit, deg(T) lt deg(A) and A*S+B*T=1. U and V are the unique elements of Z sub M (x) such that A*U+B*V=C, with deg(V) lt deg(A).}

procedure MUPBQP (PL,A: LIST): LIST;
{Modular univariate polynomial Berlekamp q polynomials construction. p is a prime beta-integer. A is a univariate polynomial over Z sub p with deg(A) ge 2. Q is the list (Q(0), ...,Q(n-1)), where Q(i)(x)= rem(x**(p*i),A(x)) and n=deg(A).}

procedure MUPDDF (PL,A: LIST): LIST;
{Modular univariate polynomial distinct degree factorization. p is a prime beta-integer. A is a monic squarefree univariate polynomial over Z sub p, with deg(A) ge 2. L is a list ((n(1),A(1)), ... ,(n(k),A(k))) where each n(i) is a positive integer, n(1) lt n(2) lt ...lt n(k), and A(i) is the product of all irreducible monic factors of A of degree n(i).}

procedure MUPFBL (PL,A: LIST): LIST;
{Modular univariate polynomial factorization-Berlekamp algorithm. p is a prime beta-integer. A is a monic squarefree univariate poly- nomial, with deg(A) ge 2. L is a list of all the monic irreducible factors of A of positive degree.}

procedure MUPFS (PL,A,B,DL: LIST): LIST;
{Modular univariate polynomial factorization, special. p is a prime beta-integer. A is a monic squarefree polynomial in Z sub p(x),deg(A) ge 2. B is a list (B(1), ...,B(r)) of monic univariate polynomials over Z sub p, which constitute a basis for the space of all polynomials C of degree less than deg(A) such that A is a divisor of C**p-C. Further-more, B(1)=1. d is a positive beta-integer such that A has no irreducible factor of degree less than d. L is a list consisting of all the monic irreducible factors of A of positive degree.}

10.11 SAC Polynomial Factorization

procedure IPCEVP (RL,A: LIST; VAR B,L: LIST);
{Integral polynomial, choice of evaluation points. A is an r-variate integral polynomial, square-

free in its main variable x, r ge 1. L is a list (l sub 1, ...,l sub r-1) of beta-integers, with L as small as possible in reverse lexicographic order with 0 lt 1 lt -1 lt 2 lt -2 lt ... such that deg sub x of A(x1, ...,x sub r-1,x)=deg sub x of A(l sub 1, ...,l sub r-1,x), and A(l sub 1, ...,l sub r-1,x) is the square- free univariate integral polynomial B.}

procedure IPFAC (RL,A: LIST; VAR SL,CL,L: LIST);
{Integral polynomial factorization. A is a non-zero integral polynomial in r variables, r ge 1. s=sign(A). c is the integer content of A. L is a list ((e1,A1), ...,(ek,Ak)), k ge 0, where each ei is a positive integer, the Aips are the distinct positive irreducible integral factors of A, and A=s*c*(the product from i equal 1 to k of Ai**ei).}

procedure IPGFCB (RL,A: LIST): LIST;
{Integral polynomial Gelfond factor coefficient bound. A is an integral polynomial in r variables, r gt 0. a=2**h*(the degree of A in xr) where h=the least integer greater than the sum from i=1 to r of the maximum of 0 and ((2*the i-th partial derivative of A)-1)/2. a is an integer.}

procedure IPIQH (RL,PL,D,AB,BB,SB,TB,M,C: LIST; VAR A,B: LIST);
{Integral polynomial mod ideal quadratic Hensel lemma. D is a list of non-negative beta-integers (d sub 1, ...,d sub r-1), r ge 1. AB, BB, SB and TB belong to Z sub p (x sub 1, ...,x sub r-1,y)/(x sub 1 ** d sub 1, ...,x sub r-1 ** d sub r-1), with AB monic, AB*SB+BB*TB=1, deg sub y of AB gt 0 and p a prime beta-integer. C is an r-variate integral polynomial with AB*BB congruent to C. M, a positive integer, is equal to p**j for some positive integer j. A, B belong to Z sub M (x sub 1, ...,x sub r-1,y)/(x sub 1 ** d sub 1, ...,x sub r-1 ** d sub r-1), with A monic, A congruent to AB, B congruent to BB, deg sub y of A=deg sub y of AB, and A*B congruent to C.}

procedure ISFPF (RL,A: LIST): LIST;
{Integral squarefree polynomial factorization. A is a positive integral polynomial in r variables, r ge 1, which with respect to its main variable is of positive degree, primitive, and squarefree. L is a list (A1, ...,Ak) of the distinct positive irreducible factors of A.}

procedure MIPISE (RL,M,D,A,B,S,T,C: LIST; VAR U,V: LIST);
{Modular integral polynomial mod ideal, solution of equation. D is a list (d sub 1, ...,d sub r-1) of non-negative beta-integers, r ge 1. A, B, S, T and C belong to Z sub M (x sub 1, ...,x sub r-1,y)/(x sub 1 ** d sub 1, ...,x sub r-1 ** d sub r-1), with A monic and of positive degree in y, and A*S+B*T=1. U and V belong to Z sub M(x sub 1, ...,x sub r-1,y)/(x sub 1 ** d sub 1, ...,x sub r-1 ** d sub r-1) such that A*U+B*V=C, and deg sub y of V lt deg sub y of A.}

procedure MPIQH (RL,PL,D,AB,BB,SB,TB,M,DP,C: LIST; VAR A,B: LIST);
{Modular polynomial mod ideal, quadratic Hensel lemma. p is a beta- prime. D and DP are lists of positive beta-integers of length r-1, r ge 1. AB, BB, SB, TB belong to Z sub p (x sub 1, ...,x sub r-1,y)/(x sub 1 ** d(1), ...,x sub r-1 ** d(r-1)), with AB monic, AB*SB+BB*TB=1, and deg sub y of AB gt 0. C is an r-variate integral polynomial, with AB* BB congruent to C. M, a positive integer, is equal to p**j for some positive integer j. A, b belong to Z sub M(x sub 1, ...,x sub r-1,y)/ (x sub 1 ** DP(1), ...,x sub rl-1 ** DP(r-1)), with A monic, A congruent to AB, B congruent to BB, deg sub y of A=deg sub y of AB, and A*B congruent to C.}

procedure MPIQHL (RL,PL,F,M,D,C: LIST): LIST;
{Modular polynomial mod ideal quadratic Hensel lemma, list. C is an r-variate integral polynomial. F is a list (f sub 1, ...,f sub m) of monic univariate polynomials of positive degree over Z sub p, with the product of the f sub i (x) similar to C(0, ...,0,x), and gcd(f sub i, f sub j)=1 for 1 le i lt j le m, p a beta-prime not dividing ldcf(C). M is a positive power of p. D is a list (d sub 1, ...,d sub r-1) of non-negative beta-integers. FP is a list (fp sub 1, ...,fp sub m) of monic polynomials in Z sub M (x sub 1, ...,x sub r-1,x)/(x sub 1 ** d sub 1, ...,x sub r-1 ** d sub r-1), with C similar to the product of the fp sub i, fp sub i congruent to f sub i, and deg sub x of fp sub i=deg sub x of f sub i, for 1 le i le m.}

procedure MPIQHS (RL,M,D,AB,BB,SB,TB,SL,NL,C: LIST; VAR A,B,S,T,DP: LIST);
{Modular polynomial mod ideal, quadratic Hensel lemma on a single variable. M is a positive

integer. D is a list of positive beta-integers (d sub 1, ...,d sub r-1), r ge 2. AB, BB, SB, TB belong to Z sub M(x sub 1, ...,x sub r-1,y)/(x sub 1 ** d sub 1, ...,x sub r-1 **d sub r-1). s is a positive integer lt r, and N is a non-negative beta-integer. C is an element of Z sub M (x sub 1, ..., x sub r-1,yl). AB is monic. AB*SB+BB*TB=1, AB*BB is congruent to C, and deg sub y of AB gt 0. A, B, S, T belong to Z sub M(x sub 1, ...,x sub r-1,y)/(x sub 1 ** d sub 1, ...,S sub s-1 ** d sub s-1,x sub s ** n,x sub s+1 ** d sub s+1, ...,x sub r-1 ** d sub r-1), with A*S+B*T=1, deg sub y of A=deg sub y of AB, A monic, A*B congruent to C, and A congruent to AB, B congruent to BB, S congruent to SB, T congruent to TB. DP is a list of non-negative beta-integers (d sub 1, ...,d sub s-1,n,d sub s+1, ...,d sub r-1).}

10.12 SAC Polynomial GCD and RES System

procedure IPC (RL,A: LIST): LIST;
{Integral polynomial content. A is an integral polynomial in r variables. C is the content of A.}

procedure IPCPP (RL,A: LIST; VAR C,AB: LIST);
{Integral polynomial content and primitive part. A is an integral polynomial in r variables. C is the content of A and AB is the primitive part of A.}

procedure IPGCDC (RL,A,B: LIST; VAR C,AB,BB: LIST);
{Integral polynomial greatest common divisor and cofactors. A and B are integral polynomials in r variables, r ge 0. C=GCD(A,B). If C is non-zero then AB=A/C and BB=B/C. Otherwise AB=0 and BB=0.}

procedure IPIC (RL,A: LIST): LIST;
{Integral polynomial integer content. A is an integral polynomial in r variables. c is the integer content of A.}

procedure IPICPP (RL,A: LIST; VAR AL,AB: LIST);
{Integral polynomial integer content and primitive part. A is an integral polynomial in r variables. a is the integer content of A. AB=A/a if A is non-zero and AB=0 if A=0.}

procedure IPICS (RL,A,CL: LIST): LIST;
{Integral polynomial integer content subroutine. A is a non-zero integral polynomial in r variables. c is an integer. d is the greatest common divisor of c and the integer content of A.}

procedure IPIPP (RL,A: LIST): LIST;
{Integral polynomial integer primitive part. A is an integral polynomial in r variables. If A ne 0 then AB=A/a where a is the integer content of A. If A=0 then AB=0.}

procedure IPPGSD (RL,A: LIST): LIST;
{Integral polynomial primitive greatest squarefree divisor. A is an integral polynomial in r variables. If A=0 then B=0. Otherwise B is the greatest squarefree divisor of the primitive part of A.}

procedure IPPP (RL,A: LIST): LIST;
{Integral polynomial primitive part. A is an integral polynomial in r variables. AB is the primitive part of A.}

procedure IPRES (RL,A,B: LIST): LIST;
{Integral polynomial resultant. A and B are integral polynomials in r variables, r ge 1, of positive degrees. C is the resultant of A and B with respect to the main variable, an integral polynomial in r-1 variables.}

procedure IPRPRS (RL,A,B: LIST): LIST;
{Integral polynomial reduced polynomial remainder sequence. A and B are non-zero integral polynomials in r variables with deg(A) ge deg(B). S is the reduced polynomial remainder sequence of A and B.}

procedure IPSCPP (RL,A: LIST; VAR SL,C,AB: LIST);
{Integral polynomial sign, content, and primitive part. A is an integral polynomial in R ge 1 variables. s is the sign of A, C is the content of A, and AB is the primitive part of A.}

procedure IPSF (RL,A: LIST): LIST;
{Integral polynomial squarefree factorization. A is a positive pri- mitive integral polynomial in r variables of positive degree. L is the list ((e sub 1,A sub 1), ...,(e sub l,A sub l)) where A equal to the product of (A sub i)**(e sub i) for i = 1, ...,k is the squarefree factorization of A in which 1 le e sub 1 lt e sub 2 lt ... lt e sub k and each A sub i is a positive squarefree polynomial of positive degree.}

procedure IPSPRS (RL,A,B: LIST): LIST;
{Integral polynomial subresultant polynomial remainder sequence. A and B are non-zero integral polynomials in r variables with deg(A) ge deg(B). S is the subresultant p.r.s. of the first kind of A and B.}

procedure IPSRP (RL,A: LIST; VAR AL,AB: LIST);
{Integral polynomial similiar to rational polynomial. A is a rational polynomial in r variables, r ge 0. a is a rational number, and AB is an integral polynomial such that A=a*AB. If A eq 0 then a=AB=0. Otherwise AB is integer primitive and positive.}

procedure MPGCDC (RL,PL,A,B: LIST; VAR C,AB,BB: LIST);
{Modular polynomial greatest common divisor and cofactors. p is a prime beta-integer. A and B are polynomials in r variables over Z sub p. C=gcd(A,B). If C is non-zero then AB=A/C and BB=B/C. Otherwise AB=0 and BB=0.}

procedure MPRES (RL,PL,A,B: LIST): LIST;
{Modular polynomial resultant. p is a prime beta-digit. A and B are polynomials over Z sub p in r variables, r ge 1, of positive degree. C is the resultant of A and B, a polynomial over Z sub p in r-1 variables.}

procedure MPSPRS (RL,PL,A,B: LIST): LIST;
{Modular polynomial subresultant polynomial remainder sequence. A and B are non-zero polynomials in r variables over Z sub p, p a prime beta-integer, with deg(A) ge deg(B). S is the subresultant p.r.s. of the first kind of A and B.}

procedure MPUC (RL,PL,A: LIST): LIST;
{Modular polynomial univariate content. A is a polynomial in r variables, r ge 2, over Z sub p, p a prime beta-integer. c is the univariate content of A.}

procedure MPUCPP (RL,PL,A: LIST; VAR AL,AB: LIST);
{Modular polynomial univariate content and primitive part. A is a polynomial in r variables, r ge 2, over Z sub p, p a prime beta-integer. a is the univariate content of A. AB=A/a if A is non-zero and AB=0 if A=0.}

procedure MPUCS (RL,PL,A,CL: LIST): LIST;
{Modular polynomial univariate content subroutine. A is a non-zero polynomial in r variables, r ge 2, over Z sub p, p a prime beta-integer. c is a univariate polynomial over Z sub p. d is the greatest common divisor of c and the univariate content of A.}

procedure MPUPP (RL,PL,A: LIST): LIST;
{Modular polynomial univariate primitive part. A is a polynomial in r variables, r ge 2, over Z sub p, p a prime beta-integer. If A is non-zero then AB=A/a where a is the univariate content of A. If A=0 then AB=0.}

procedure MUPEGC (PL,A,B: LIST; VAR C,U,V: LIST);
{Modular univariate polynomial extended greatest common divisor. p is a prime beta-integer. A and B are univariate polynomials over Z sub p. C=gcd(A,B). A*U+B*V=C, and, if deg(A/C) gt 0, then deg(V) lt deg(A/C), else deg(V)=0. Similarly, if deg(B/C) gt 0, then deg(U) lt deg(B/C), else deg(U)=0. If A=0, U=0. If B=0, V=0.}

procedure MUPGCD (PL,A,B: LIST): LIST;
{Modular univariate polynomial greatest common divisor. A and B are univariate polynomials over Z sub p, p a prime beta-integer. C=gcd(A,B).}

procedure MUPHEG (PL,A,B: LIST; VAR C,V: LIST);
{Modular univariate polynomial half-extended greatest common divisor. p is a prime beta-integer. A and B are univariate polynomials over Z sub p. C=gcd(A,B). There exists a polynomial U such that A*U+B*V=C, and, if deg(A/C) gt 0, then deg(V) lt deg(A/C). If deg(A/C)=0, deg(V) is also 0. If B=0, V=0.}

procedure MUPRES (PL,A,B: LIST): LIST;
{Modular univariate polynomial resultant. p is a prime beta-digit. A and B are univariate polynomials over Z sub p of positive degrees. C is the resultant of A and B, an element of Z sub p.}

procedure MUPSFF (PL,A: LIST): LIST;
{Modular univariate polynomial squarefree factorization. p is a prime beta-integer. A is a monic univariate polynomial over Z sub p of positive degree. L is a list ((i(1),A(1)), ...,(i(r),A(r))) with i(1) lt i(2) lt ... lt i(r), A(j) a monic squarefree factor of a of positive degree for 1 le j le r and A the product of A(j)**i(j) for j=1, ...,r.}

procedure RPBLGS (RL,A: LIST; VAR AL,BL,SL: LIST);
{Rational polynomial base coefficients least common multiple, greatest common divisor, and sign. A is a rational polynomial in r variables, r ge 0. If A=0 then a=b=s=0. Otherwise, a is the lcm of the denominators of the base coefficients of A, b is the gcd of the numerators of the base coefficients of A, and s is the sign of the leading base coefficient of A.}

10.13 SAC Polynomial Real Root

procedure IIC (A,AP,I,L: LIST): LIST;
{Isolating interval conversion. A is a squarefree univariate integral polynomial. AP is the derivative of A. I is an left open right closed interval (a,b) with binary rational endpoints represented by the list (a,b). L is a list of isolating intervals with binary rational end- points for the real roots of A in I. L=((a(1),b(1)), ...,(a(k),b(k))) with a(1) le b(1) le ... le a(k) le b(k) and (a(i), b(i)) represents the open interval (a(i),b(i)) if a(i) lt b(i), the closed interval (a(i),b(i)) if a(i)=b(i). LS is a list ((as(1),bs(1)), ...,(as(k),bs(k))) of isolating intervals for the same roots and satisfying the same conditions except that each pair (as(i),bs(i)) represents the left open right closed interval (as(i),bs(i)).}

procedure IPFSD (RL,A: LIST): LIST;
{Integral polynomial factorization, second derivative. A is a positive primitive integral polynomial in r variables of positive degree. L is a list (a(1), ...,a(k)) where k ge 1, A equal to sum of a(i) for i=1, ...,k and, for each i, a(i) is a positive primitive integral polynomial of positive degree with deg(a(i)) le 2 or gcd(a(i),app(i))=1 where app(i) is the second derivative of a(i).}

procedure IPLRRI (L: LIST): LIST;
{Integral polynomial list real root isolation. L is a non-empty list (A sub 1 , ..., A sub n) of distinct univariate integral polynomials which are positive, of positive degree, squarefree, and pairwise relatively prime. M is a list (I sub 1 , B sub 1 , ..., I sub m , B sub m), where I sub 1 lt I sub 2 lt ... lt I sub m are strongly disjoint isolating intervals for all of the real roots of A eq prod from (j eq 1) to n (A sub j). Each I sub i has binary rational number endpoints, and is either left-open and right-closed or is a one-point interval. B sub i is the unique A sub j which has a root in I sub i.}

procedure IPRCH (A,I,KL: LIST): LIST;
{Integral polynomial real root calculation, high precision. A is a univariate integral polynomial of positive degree. I is either the nulllist () or a standard interval or an interval whose positive

and non-positive parts are standard. k is a gamma-integer. L is a list ((e(1),I(1)), ...,(e(r),I(r))) where the e(j) are beta-integers, 1 le e(1) le ... le e(r), and the I(j) are binary rational isolating intervals, I(j)=(a(j),b(j)), for the r distinct real roots of A if I=(), or for the r distinct real roots of A in I if I ne (). e(j) is the multiplicity of the root alpha(j) in I(j) and abs(b(j)-a(j)) le 2**k. I(j) is a left-open and right-closed interval if a(j) lt b(j), a one-point interval if a(j)=b(j).}

procedure IPRCHS (A,I,KL: LIST): LIST;
{Integral polynomial real root calculation, high-precision special. A is a positive, primitive, squarefree, univariate, integral polynomial of positive degrre with GCD(A,APP)=1. I is either the null list () or a standard interval or an interval whose positive and non-positive parts are standard. k is a gamma-integer. L is a list (I(1), ...,I(r)) of binary rational isolating intervals I(j)=(a(j),b(j)) for the r distinct real roots of A if I=(), for the r distinct real roots of A of I if I ne (), with b(j)-a(j) le 2**kl. I(j) is a left-open and right-closed interval if a(j) ne b(j), a one-point interval if a(j)=b(j).}

procedure IPRCNP (A,I: LIST; VAR SLP,SLPP,J: LIST);
{Integral polynomial real root calculation, newton method preparation. A is a positive, primitive, squarefree, univariate integral polynomial of positive degree. I is an open interval (a1,a2) with binary rational endpoints containing no roots of AP and APP. sp and spp, beta-integers, are the signs of AP and APP on I. J is a subinterval (b1,b2) of I with binary rational endpoints, containing alpha and such that spp*SIGN(AP(b1)+f*AP(b2)) ge 0, where f=(-3/4)**(sp*spp). J is a left-open and right-closed interval if b1 lt b2, the one-point interval if b1=b2.}

procedure IPRCN1 (A,I,SL,KL: LIST): LIST;
{Integral polynomial real root calculation, 1 root. A is a positive primitive univariate integral polynomial of positive degree. I is an open interval (a1,a2) with binary rational endpoints containing a unique root alpha of A and containing no roots of AP or APP. s, a beta-integer, is the sign of AP*APP on I. min(abs(AP(a1)),abs(AP(a2))) le (3/4)*max(abs(AP(a1)),abs(AP(a2))). k is a beta-integer. J is a subinterval of I of length 2**k or less containing alpha and with binary rational endpoints.}

procedure IPRIM (A: LIST): LIST;
{Integral polynomial real root isolation, modified Uspensky method. A is a non-zero squarefree univariate integral polynomial. L is a list (I sub 1 , ..., I sub r) of strongly disjoint isolating intervals for all of the real roots of A with I sub 1 lt I sub 2 lt ... lt I sub r. Each I sub j has binary rational endpoints and is either left-open and right-closed or a one-point interval.}

procedure IPRIMO (A,AP,I: LIST): LIST;
{Integral polynomial real root isolation, modified Uspensky method, open interval. A is a univariate integral polynomial without multiple roots. AP is the derivative of A. I is an open interval (a,b) with binary rational endpoints, represented by the list (a,b), such that there are integers h and k for which a=h*2**k and b=(h+1)*2**k. L is a list (I(1), ...,I(r)) of isolating intervals for the real roots of A in I. Each I(j) is a left open right closed interval with binary rational endpoints and I(1) lt I(2) lt ... lt I(r).}

procedure IPRIMS (A,AP,I: LIST): LIST;
{Integral polynomial real root isolation, modified Uspensky method, standard interval. A is a univariate integral polynomial without multiple roots. AP is the derivative of A. I is a standard interval. L is a list (I(1), ...,I(r)) of isolating intervals for the real roots of A in I. Each interval I(j) is a left open right closed interval (a(j),b(j)) with binary rational endpoints and I(1) lt I(2) lt ... lt I(r).}

procedure IPRIMU (A: LIST): LIST;
{Integral polynomial real root isolation, modified Uspensky method, unit interval. A is a squarefree univariate integral polynomial. L is a list (I(1), ...,I(r)) of isolating intervals for all the roots of A in the left closed right open interval (0,1). Each I(j) is a pair (a(j),b(j)) of binary rational numbers, with 0 le a(1) le b(1) le ... le a(r) le b(r). If a(j)=b(j) then (a(j),b(j)) represents the one-point interval (a(j),b(j)). If a(j) lt b(j) then the pair (a(j),b(j)) represents the open interval

(a(j),b(j)).}

procedure IPRIU (A: LIST): LIST;
{Integral polynomial real root isolation, Uspensky method. A is a non-zero squarefree univariate integral polynomial. L is a list of pairs ((a(1),b(1)), ...,(a(k),b(k))) representing isolating intervals forall of the real roots of A, with a(1) le b(1) le ... le a(k) le b(k). If a(i) lt b(i) then the pair (a(i),b(i)) represents the open interval (a(i),b(i)), while if a(i)=b(i) then it represents the closed one-point interval (a(i),b(i)). The a(i) and b(i) are rational numbers except that one may have a(1) equal to negative infinity, represented by -1/0, that is the pair (-1,0), and one may have b(k) equal to infinity, represented by 1/0.}

procedure IPRIUP (A: LIST): LIST;
{Integral polynomial real root isolation, Uspensky method, positive roots. A is a non-zero squarefree univariate integral polynomial. L is a list of pairs ((a(1),b(1)), ...,(a(k),b(k))) representing isolating intervals for the positive real roots of A, with a(1) le b(1) le ... le a(k) le b(k). If a(i) lt e(i) then the pair (a(i), b(i)) represents the open interval (a(i),b(i)) while if a(i)=b(i) then (a(i),b(i)) represents the closed one-point interval (a(i),b(i)). The a(i) and b(i) are rational numbers except thatone may have b(k) equal to infinity, represented by 1/0, that is, the pair (1,0).}

procedure IPRRLS (A1,A2,L1,L2: LIST; VAR LS1,LS2: LIST);
{Integral polynomial real root list separation. A1 and A2 are univariate integral polynomials with no multiple real roots and with no common real roots. L1 and L2 are lists of isolating intervals for some or all of the real roots of A1 and A2, respectively. The intervals in L1 and L2 have binary rational endpoints, and are either left-open and right-closed or one-point intervals. Let L1 eq (I sub 1,1 , ..., I sub (1,r sub 1)), L2 eq (I sub 2,1 , ..., I sub (2,r sub 2)). Then I sub 1,1 lt I sub 1,2 lt ... lt I sub (1,r sub 1) and I sub 2,1 lt I sub 2,2 lt ... lt I sub (2,r sub 2) . L sub 1 sup * eq (I sub 1,1 sup * , ..., I sub (1,r sub 1) sup *) and L sub 2 sup * eq (I sub 2,1 sup * , ..., I sub (2,r sub 2) sup *), where I sub i,j sup * is a binary rational subinterval of I sub i,j containing the root of A sub i in I sub i,j. Each I sub 1,j sup * is strongly disjoint from each I sub 2,j sup *.}

procedure IPRRS (A1,A2,I1,I2: LIST; VAR IS1,IS2,SL: LIST);
{Integral polynomial real root separation. A1 and A2 are squarefree univariate integral polynomials of positive degrees. I1 and I2 are intervals with binary rational number endpoints, each of which is either left-open and right-closed, or a one-point interval. I1 contains a unique root alpha sub 1 of A1, and I2 contains a unique root alpha sub 2 ne alpha sub 1 of A2. I sub 1 sup * and I sub 2 sup * are binary rational subintervals of I1 and I2 containing alpha sub 1 and alpha sub 2 respectively, with I sub 1 sup * and I sub 2 sup * strongly disjoint. If I1 is left-open and right-closed then so is I sub 1 sup *, and similarly for I2 and I sub 2 sup *. s eq -1 if I sub 1 sup * lt I sub 2 sup *, and s eq 1 if I sub 1 sup * gt I sub 2 sup *.}

procedure IPSFSD (RL,A: LIST): LIST;
{Integral squarefree factorization, second derivative. A is a positive integral polynomial in r variables of positive degree L is a list ((e(1),A(1)), ...,(e(k),A(k))) where primitive part of A is equal to the sum of A(i)**e(i) for i=1, ...,k. The a(i) are pairwise relatively prime squarefree positive polynomials of positive degrees, with deg(A(i))=1 or GCD(A(i),APP(i))=1 for all i where APP(i) is the second derivative of A(i) and the e(i) are positive beta-integers e(1) le e(2) le ... le e(k).}

procedure IPSRM (A,I: LIST): LIST;
{Integral polynomial strong real root isolation, modified Uspensky method. A is a univariate integral polynomial with multiple roots and with no real roots in common with APP. I is either the null list () or a standard interval or an interval whose positive and non-negative parts are standard. L is a list (I(1), ...,I(r)) of isolating intervals for all the real roots of A if I=(), or for all the real roots of A in I if I ne (). The intervals I(j) contain no roots of AP or APP, are left-open and right-closed, have binary rational endpoints, and satisfy I(1) lt I(2) lt ... lt I(r).}

procedure IPSRMS (A,I: LIST): LIST;
{Integral polynomial strong real root isolation, modified Uspensky method, standard interval. A

is a univariate integral polynomial with no multiple real roots and with no real roots in common with APP. I is a standard interval. L is a list (I(1), ...,I(r)) of isolating intervals for the roots of A in I. The intervals I(j) contain no roots of AP or APP, are left-open and right-closed, have binary rational endpoints, are subintervals of I, and satisfy I(1) lt I(2) lt ... lt I(r).}

procedure IPVCHT (A: LIST): LIST;
{Integral polynomial variations after circle to half-plane transformation. A is a non-zero univariate integral polynomial. Let n=deg(A), AP(x)=(x**n)*A(1/x), B(x)=AP(x+1). k is the number of sign variations in the coefficients of B.}

procedure IUPRB (A: LIST): LIST;
{Integral univariate polynomial root bound. A is an integral poly- nomial of positive degree. b is a binary rational number which is a root bound for A. If A(x) is equal to the sum of a(i)*x(i)**i for i=0, ...,n, a(n) ne 0, then b is the smallest power of 2 such that 2*abs(a(n-k)/a(n))**(1/k) le b for 1 le k le n. If a(n-k)=0 for 1 le k le n then b=1.}

procedure IUPRLP (A: LIST): LIST;
{Integral univariate polynomial, root of a linear polynomial. A is an integral univariate polynomial of degree one. r is the unique rational number such that A(r)=0.}

procedure IUPVAR (A: LIST): LIST;
{Integral univariate polynomial variations. A is a non-zero uni- variate integral polynomial. n is the number of sign variations in the coefficients of A.}

procedure IUPVSI (A,I: LIST): LIST;
{Integral univariate polynomial, variations for standard interval. A is a non-zero integral univariate polynomial. I is a standard open interval. Let T(z) be the transformation mapping the right half-plane onto the circle having I as a diameter. Let B(x)=A(T(x)). Then v is the number of sign variations in the coefficients of B.}

procedure RIB (RL,SL: LIST): LIST;
{Rational interval bisection. r and s are binary rational numbers, r lt s. t is a binary rational number with r lt t lt s, defined as follows. Let h=floor(log2(s-r)) and let c be the least integer such that c*2**h gt r. Then t=c*2**h if c*2**h lt s and t=(2*c-1)*2**(h-1) otherwise.}

procedure RILC (I,KL: LIST): LIST;
{Rational interval length comparison. I is an interval (a,b) with rational endpoints, a le b. k is a gamma-integer. t=1 if b-a le 2**k and t=0 otherwise.}

procedure RINT (I: LIST): LIST;
{Rational interval normalizing transformation. I is a list (r,s) with rational endpoints and r lt s. IS is the list (rs,ss)= psi(r,s).}

10.14 SAC Univariate Polynomial Factorization

procedure IPFLC (RL,M,I,A,L,D: LIST): LIST;
{Integral polynomial factor list combine. A is a non-constant primitive r-variate integral polynomial. M is a positive integer. I is a list (d sub 1, ...,d sub r - 1) of non-negative beta-digits. L is a list of monic factors of A modulo M, ((x sub 1)**(d sub 1), ... ,(x sub rl- 1)**(d sub r - 1)) such that if B is an integral factor of A, then H sub M,I (B) is an associate of some product of elements of L. D is either 0, or a characteristic set for the possible degrees of integral factors of A. LP is a list of the primitive irreducible integral factors of A. }

procedure IUPFAC (A: LIST; VAR SL,CL,L: LIST);
{Integral univariate polynomial factorization. A is a non-zero integral univariate polynomial. s=sign(A), c=cont(A). L is a list ((e1,A1), ...,(ek,Ak)), k ge 0, where each ei is a positive integer, e1 le e2 le ... le ek, each A i is an ir- reducible positive integral univariate polynomial, and A = s * c * the product of A i ** ei, 1 le i le k.}

procedure IUPFDS (A: LIST; VAR PL,F,C: LIST);
{Integral univariate polynomial factor degree set. A is a non-zero square-free integral polynomial. C is the intersection of the degree sets of factorizations over Z sub p for as many as NPFDS primes p (fewer only if SMPRM is exhausted or A is proved irredu- cible). C is represented as a characteristic set. p is the least examined prime in P which gave the smallest number of factors, and F is the distinct degree factorization of A over Z sub p, unless A is shown to be irreducible, in which case p=0, F=().}

procedure IUPQH (PL,AB,BB,SB,TB,M,C: LIST; VAR A,B: LIST);
{Integral univariate polynomial quadratic Hensel lemma. AB, BB, SB, TB are univariate polynomials over Z sub p, p a prime beta-integer, with AB*SB+BB*TB=1, and deg(TB) lt deg(AB). C is a univariate integral polynomial with H sub p of C=AB*BB. M, a positive integer, is equal to p**j for some positive integer j. A and B are univariate polynomials over Z sub M, with H sub p of A=AB, H sub p of B=BB, ldcf(A)=ldcf(AB),deg(A)=deg(AB), and H sub M of C=A*B.}

procedure IUPQHL (PL,F,M,C: LIST): LIST;
{Integral univariate polynomial quadratic Hensel lemma, list. C is an integral univariate polynomial. F is a list (f sub 1, ...,f sub r) of monic polynomials in Z sub p (x) with H sub p of C similar to the product of the f sub i, and gcd(f sub i,f sub j)=1 for 1 le i lt j le r, p a beta-prime not dividing ldcf(C). M is a positive power of p. FP is a list (fp sub 1, ...,fp sub r) of monic polynomials in Z sub M (x) with H sub M of C similar to the product of the fp sub i, H sub p of fp sub i=f sub i and deg(fp sub i)=deg(f sub i), for 1 le i le r.}

procedure IUSFPF (A: LIST): LIST;
{Integral univariate squarefree polynomial factorization. A is an integral univariate squarefree polynomial which is positive, primitive and of positive degree. L is a list (A1, ...,Ak) of the positive irreducible factors of A.}

Chapter 11

Non-commutative Algorithms

11.1 DIP Groebner bases for non noetherian polynomial rings.

procedure DINNCP (EL,A,B: LIST): LIST;
{distributive polynomial non-commutative product. e is a non-negative integer. A and B are distributive polynomials in 2 variables. C is the non-commutative product of A and B with respect to Y * X = X**e Y. }

procedure EVNNCP (EL,S,T: LIST): LIST;
{exponent vector non-commutative product. S and T are exponent vectors. of length 2. C is the non-commutative product S * T with respect to the relation Y * X = X**e Y. }

procedure EVNRDT (EL,S,T:LIST):LIST;
{exponent vectors non-commutative right division test. s and t are exponent vectors. C=1 if s rdiv t otherwise, C=0. }

procedure EVNCRD (EL,S,T:LIST):LIST;
{exponent vectors non-commutative right division. s and t a are exponent vectors, if s rdiv t then the output is t//s. }

procedure EVNLDT (EL,S,T:LIST):LIST;
{exponent vectors non-commutative left division test. s and t are exponent vectors. C=1 if s ldiv t otherwise, C=0. }

procedure EVNCLD (EL,S,T:LIST):LIST;
{exponent vectors non-commutative left division. s and t a are exponent vectors. The output is s t if s ldiv t . }

procedure EVLLCM (EL,S,T:LIST):LIST;
{exponent vectors non-commutative least left common multiple. s and t are exponent vectors. C is the least left common multiple of s and t. }

procedure EVLRCM (EL,S,T:LIST):LIST;
{exponent vectors non-commutative least right common multiple. s and t exponent vectors. C is the least right common multiple of s and t if it exists! }

procedure EVRCMT (EL,S,T:LIST):LIST;
{exponent vectors non-commutative right multiple test. S and T are exponent vectors. C=1 if S and T have some right common multiple otherweise, C=0. }

procedure DNNLNF (EL,P,S:LIST):LIST;
{distributive polynomials non-noetherian left normal form. P is a list of non zero polynomials in distributive rational representation. S is a distributive rational polynomial. R is a polynomial such that S is left reducible to R modulo P and R is in normal form with respect to P. }

procedure DNNLIS (EL,P: LIST): LIST;
{distributive polynomials non-noetherian left irreducible set. P is a list of distributive rational polynomials, PP is the result of left reducing each p element of P modulo P-(p) until no further reductions are possible. }

procedure DNNLSP (EL,A,B:LIST):LIST;
{distributive polynomials non-noetherian left S-polynomial. A and B are polynomials in distributive rational representation in 2 variables. C is the left S-polynomial of A and B. }

procedure DNNLGB (EL,P,TF: LIST): LIST;
{distributive non-noetherian polynomials left Groebner base. P is a list of rational polynomials in distributive representation in 2 variables. PP is the left Groebner base of P. t is the trace flag. }

procedure DNNRNF (EL,P,S:LIST):LIST;
{distributive polynomials non-noetherian right normal form. P is a list of non zero polynomials in distributive rational representation. S is a distributive rational polynomial. R is a polynomial such that S is right reducible to R modulo P and R is in normal form with respect to P. }

procedure DNNRIS (EL,P: LIST): LIST;
{distributive polynomials non-noetherian right irreducible set. P is a list of distributive rational polynomials, PP is the result of right reducing each p element of P modulo P-(p) until no further reductions are possible. }

procedure DNNRSP (EL,A,B:LIST):LIST;
{distributive polynomials non-noetherian right S-polynomial. A and B are polynomials in distributive rational representation. C is the right S-polynomial of A and B if it exists! }

procedure DNNRGB (EL,P,TF: LIST): LIST;
{distributive polynomials non-noetherian right Groebner base. P is a list of rational polynomials in distributive representation in 2 variables. PP is the right Groebner base of P. t is the trace flag. }

procedure DNLCPL (EL,P: LIST; VAR D,B: LIST);
{distributive polynomial non-noetherian left construct pair list. P is list of polynomials in distributive representation in 2 variables. B is the polynomials pairs list and D is the pairs list. }

procedure DNRCPL (EL,P: LIST; VAR D,B: LIST);
{distributive polynomial non-noetherian right construct pair list. P is list of polynomials in distributive representation in 2 variables. B is the polynomials pairs list and D is the pairs list. }

procedure DNLUPL (EL,PL,P,D,B: LIST): LIST;
{distributive polynomial non-noetherian left update pair list. P is list of polynomials in distributive representation in 2 variables. B is the polynomials pairs list and D is the pairs list. p is a non zero polynomial in distributive representation. D, P and B are modified. DP is the updated pairs list. }

procedure DNRUPL (EL,PL,P,D,B: LIST): LIST;
{distributive polynomial non-noetherian right update pair list. P is list of polynomials in distributive representation in 2 variables. B is the polynomials pairs list and D is the pairs list. p is a non zero polynomial in distributive representation. D, P and B are modified. DP is the updated pairs list. }

procedure DNN2GB (EL,P,TF: LIST): LIST;
{distributive polynomials non-noetherian 2-sided Groebner base. P is a list of rational polynomials

in distributive representation in 2 variables. PP is the Groebner base of P. t is the trace flag.}

procedure DNNTGB (EL,P,TF:LIST):LIST;
{distributive polynomials non-noetherian two-sided Groebner base. P is a list of rational polynomials in distributive representation in 2 variables. PP is the two-sided Groebner base of P. t is the trace flag. The non-commutative produkt is computed w.r.t Y*X=X**eY. }

procedure DNNLES (EL,P:LIST):LIST;
{distributive polynomials non-noetherian left exponents set. P is a list of polinomials in distributive representation. PP is the the list which is result of *-multiplication of each polynomial of P from the left with the main variable. The non-commutative multiplication is computed w.r.t the relation Y * X = X**e Y. }

procedure DNNRES (EL,P,DL,DP:LIST):LIST;
{distributive polynomials non-noetherian right exponents set. P is a list of polynomials in distributive representation, d and d'are non- negative integers with d' ¿= d. PP is the is result of *-multiplication of each polynomial of P from the right with exponents (e**i) of the first variable in the variable list, where i ranges from d to d'. The *-multiplication is computed w.r.t the relation Y * X = X**e Y. }

procedure DIPSPS (D,B:LIST):LIST;
{distibutive polynomials S-polynomials set. D is the pairs list and B is the polynomials pairs list. D and B are modified. H is the set of all non-zero S-polynomials. }

procedure DIPLMD (P:LIST):LIST;
{distributive polynomial list maximum degree. P is a non-empty list of polynomials in distributive form in r variables. d is the maximum degree of all polynomials of P w.r.t the main variable .}

procedure IPOWER (EL,AL:LIST):LIST;
{integer power. e and a are positive integers. C=e**a. }

11.2 DIP Exterior Algebra

procedure COPYOB (A: LIST): LIST;
{Copy object. A is an object. B is the copy of A.}

procedure EIMWRT (A: LIST);
{Exterior integral matrix write. A is an exterior integral matrix. A is written in the output stream.}

procedure EIVABS (U: LIST): LIST;
{Exterior integral vector absolute value. U is an exterior integral vector. V is the absolute value of U. }

procedure EIVAPP (U: LIST): LIST;
{Exterior integral vector absolute primitive part. U is an exterior integral vector. V is the absolute primitive part of U. }

procedure EIVCPP (U: LIST; VAR V,VL: LIST);
{Exterior integral vector content and primitive part. U is an exterior integral vector. v is the content and V is the primitive part of U. }

procedure EIVEPR (U,V: LIST): LIST;
{Exterior integral vector exterior product. U and V are exterior integral vectors. W is the exterior product of U and V.}

procedure EIVFUP (A,PL: LIST): LIST;
{Exterior integral vector from univariate integral polynomial with multiplication by power of main variable. A is an univariate integral polynomial. p is a beta-integer. B is the exterior integral vector from A(x)*(x**p). }

procedure EIVILP (U,V: LIST): LIST;
{Exterior integral vector inner left product. U and V are exterior integral vectors. W is the inner left product of U and V.}

procedure EIVIP (A,BL: LIST): LIST;
{Exterior integral vector integer product. A is an exterior integral vector, b is an integer, C=A*b. }

procedure EIVIQ (A,BL: LIST): LIST;
{Exterior integral vector integer quotient. A is an exterior integral vector, b is a nonzero integer, and b divides any coefficient of A. C=A/b.}

procedure EIVIRP (U,V: LIST): LIST;
{Exterior integral vector inner right product. U and V are exterior integral vectors. W is the inner right product of U and V.}

procedure EIVNEG (U: LIST): LIST;
{Exterior integral vector negative. U is an exterior integral vector. V is the negative of U. }

procedure EIVPP (U: LIST): LIST;
{Exterior integral vector primitive part. U is an exterior integral vector. V is the primitive part of U. }

procedure EIVSIG (U: LIST): LIST;
{Exterior integral vector sign. U is an exterior integral vector. s is the sign of U. }

procedure EIVSUM (U,V: LIST): LIST;
{Exterior integral vector sum. U and V are exterior integral vectors. W is the sum of U and V.}

procedure EIVWRT (A: LIST);
{Exterior integral vector write. A is an exterior integral vector. A is written in the output stream.}

procedure EXIDET (M: LIST): LIST;
{Exterior integral matrix determinant. M is an exterior integral matrix. d is the determinant of A.}

procedure EXIDT2 (M: LIST): LIST;
{Exterior integral matrix determinant 2. M is an exterior integral matrix. d is the determinant of A.}

procedure EXMHOM (M: LIST): LIST;
{Exterior matrix homomorphism. M=(m1,... ,mn) is a vector of integral vectors mi, 0 le i le n. MS is a vector of exterior integral vectors, MS=(ms1,... ,msn). were msi=EXVHOM(mi). }

procedure EXVHOM (U,SL: LIST): LIST;
{Exterior vector homomorphism. U=(u1,... ,un) is an integral vector of n components, 0 le n. s is the starting index for the exterior index list. V=(u1,(s),... ,un,(s+n)). }

procedure ITD (A: LIST): LIST;
{Integer trailing digit. A is an integer, A = b mod beta.}

procedure IJACS (X,Y: LIST): LIST;
{Integer Jacobi symbol algorithm. Y is an odd positive integer, X is an integer relatively prime to Y. s=(X/Y). }

procedure ILADDC (U,CL: LIST): LIST;
{Index list addition of constant. U is an index list, c is a beta-integer. V=(u1+c, ...,un+c) where U=(u1, ...,un). n ge 0. }

procedure ILEXPR (U,V: LIST; VAR W,SL: LIST);
{Index list exterior product. U, V and W are index lists. W is the exterior product of U and V. s is the sign of the exterior product. If s = 0 then W = (). }

procedure ILILPR (U,V: LIST; VAR W,SL: LIST);
{Index list inner left product. U, V and W are index lists. W is the inner left product of U and V. s is the sign of the inner left product. If s = 0 then W = (). }

procedure ILINPR (U,V: LIST; VAR W,SL: LIST);
{Index list inner product. U, V and W are index lists. W is the inner product of U and V, i.e. if U is contained in V then W is the complement of U in V, otherwise the sign of the inner product is set to zero. s is the sign of the inner product. }

procedure ILIRPR (U,V: LIST; VAR W,SL: LIST);
{Index list inner right product. U, V and W are index lists. W is the inner right product of U and V. s is the sign of the inner right product. if s = 0 then W = (). }

procedure ILSCMP (U,V: LIST): LIST;
{Index list strong compare. U=(u1,... ,un), V=(v1,... vm) are index lists with length n and m. t=1 if n gt m, t=-1 if n lt m. If n=m then t=0 if U=V, t=1 if U gt V, t=-1 if U lt V.}

procedure ILWCMP (U,V: LIST): LIST;
{Index list week compare. U=(u1,... ,un), V=(v1,... vm) are index lists. t=0 if U=V, t=1 if U gt V, t=-1 if U lt V.}

procedure INDLST (RL,SL: LIST): LIST;
{Index list. Starting with r and ending with s.}

procedure INLWRT (U: LIST);
{Index list write. U is an exterior index list. U is written in the output stream.}

procedure IPSR (R: LIST): LIST;
{Integral polynomial specified roots. R is a list of integers. A is an integral univariate polynomial with roots from R. }

procedure IVHOM (U,IL,JL: LIST): LIST;
{Integer vector homomorphism. U=(u1,(s),... ,un,(r)) is an exterior integral vector. i is the starting index for the integral vector and j is its ending index. V=(vi,... ,vj). }

procedure IVRAND (KL,QL,NL: LIST): LIST;
{Integer vector random. U is an random integer vector with n components, 0 le n, and the absolut value of each component is lt 2**k. q is a rational number qd/qn, with 0 lt qd le qn lt beta. So q is the propability that any particular component of V is not zero.}

procedure KREISP (NL: LIST): LIST;
{Kreisteilungs polynom. n is a beta-integer gt 1. A is an univariate integral polynomial. }

procedure MDVHOM (ML,U: LIST): LIST;
{Modular vector homomorphism. U is an integral vector. V is a modular vector. m is a beta-integer.}

procedure MIRAND (KL,QL,NL,ML: LIST): LIST;
{Matrix random. M is an integral matrix with n rows generated by IVRAND(k,q,m). }

procedure POWSEV (PL,A: LIST): LIST;
{Power of variable symmetric product with exterior vector. p is a beta-integer. A is an exterior vector. B is the symmetric product of x**p and A.}

procedure UIPRES (A,B: LIST; VAR CL,KL: LIST);
{Univariate integral polynomials resultant. A and B are univariate integral polynomials. c is the resultant of A and B. k is the degree of the common factor. }

procedure UIPRS1 (A,B: LIST): LIST;
{Univariate integral polynomials resultant 1. A and B are univariate integral polynomials. c is the resultant of A and B. }

procedure UIPSIL (A,EL: LIST): LIST;
{Univariate integral polynomial symmetric product with exterior index list. A is an univariate

integral polynomial. e is an exterior index list. B is the symmetric product of A and e.}

procedure UIPSIV (A,B: LIST): LIST;
{Univariate integral polynomial symmetric product with exterior integral vector. A is an univariate integral polynomial. B is an exterior integral vector. C is the symmetric product of A and B.}

11.3 MAS Non-commutative Product

procedure DINPPR (T,A,B: LIST): LIST;
{Distributive polynomial non-commutative product. A and B are distributive polynomials. T is a table of distributive polynomials specifying the non-commutative relations. C=A*B, the non-commutative product of A and B. The table T may be modified. }

procedure DINPTL (T,EL,FL: LIST; VAR C,EP,FP: LIST);
{Distributive polynomial non-commutative product table lookup. e and f are exponent vectors. T is a table of distributive polynomials specifying the non-commutative relations. C is the non-commutative product of x**es and x**fs. ep and fp are exponent vectors with es+ep=e and fs+fp=f. If e=es or f=fs then ep=() or fp=(). }

procedure DINPTU (T,EL,FL,C: LIST);
{Distributive polynomial non-commutative product table update. e and f are exponent vectors. T is a table of distributive polynomials specifying the non-commutative relations. C is a distributive rational polynomial. The relation e * f = C is added to T. T is modified. }

procedure DINPEX (T,A,NL: LIST): LIST;
{Distributive non-commutative polynomial exponentiation. A is a distributive rational polynomial, n is a non-negative beta- integer. T is a table of non-commutative relations. B=A**n. 0**0 is by definition a polynomial in zero variables. }

procedure DINLRD (V,T: LIST): LIST;
{Distributive non-commutative polynomial list read. V is a variable list. T is a table of non-commutative relations. A list of distributive non-commutative polynomials in r variables, where r=length(V), r ge 0, is read from the input stream. Any blanks preceding A are skipped. }

procedure DINPRD (V,T: LIST): LIST;
{Distributive non-commutative polynomial read. V is a variable list. T is a table on non-commutative relations. A distributive rational polynomial A in r variables, where r=length(V), r ge 0, is read from the input stream. Any blanks preceding A are skipped. }

procedure EVZERO (RL: LIST): LIST;
{Exponent vector zero. U is an exponent vector of length r, r ge 0 with all components zero. }

11.4 MAS Non-commutative Center

procedure DINCCO (T, A, B: LIST): LIST;
{Distributive rational non-commutative polynomial commutator. A and B are distributive rational non-commutative polynomials. The commutator of A and B is returned. T is the relation table. }

procedure DINCCP (T, E: LIST): LIST;
{Distributive rational non-commutative polynomial center polynomial. E is a list of exponent vectors. T is the relation table. A polynomial in the center of the ideal is returned. }

procedure DINCCPpre (T, E: LIST): LIST;
{Distributive rational non-commutative polynomial center polynomial preparation. E is a list of

exponent vectors. T is the relation table. A polynomial in the center of the polynomial ring is returned. }

procedure DILFEL (a, E: LIST): LIST;
{Distributive polynomial list from exponent vector list. E is a list of exponent vectors. A list distributive polynomials with exponent vectors from E and coefficients equal to a is returned. }

procedure DINPTslT (T: LIST): BOOLEAN;
{Distributive polynomial non-commutative product table strict lex test. T is a table of distributive polynomials specifying the non-commutative relations. It is tested if T is strict lexicographical, i.e. if Xj*Xi = cij Xi Xj + pij is a strict lexicographical commutator relation, then cij = 1 and pij ¡(inv lex) Xi Xj. }

procedure DINLMPG (T,i,F: LIST): LIST;
{Distributive non-commutative left rational minimal polynomial for a G basis. F is a non-commutative left groebner basis. PP is the left minimal polynomial for the i-th variable for F. }

procedure DINLMPL (T,F: LIST): LIST;
{Distributive non-commutative left rational minimal polynomial list for a G basis. F is a non-commutative left groebner basis. P is the list of left minimal polynomial for each variable for F. }

procedure EVGCD (U,V: LIST): LIST;
{Exponent vector greatest common divisor. U=(UL1, ...,ULRL), V=(VL1, ...,VLRL) are exponent vectors of length r. W=(WL1, ...,WLRL) is the greatest common divisor of U and V. }

procedure EVLGTD (r,d,L: LIST): LIST;
{Exponent vector list generate for total degree. r is the number of variables. L is a list of already generated exponent vectors. A list of exponent vectors up to total degree d (¿= 0) is returned. }

procedure EVLGIL (D: LIST): LIST;
{Exponent vector list generate for inverse lexicographical sequence. D is a list of maximal degrees in the respective variable. A list of exponent vectors up to the maximal degrees is returned. }

procedure EVLINV (L,i,k: LIST): LIST;
{Exponent vector list introduction of new variables. L is a list of exponent vectors. In each element of L k new variables are introduced after position i. The new list is returned. }

procedure EVTSZ (i,U: LIST): BOOLEAN;
{Exponent vector test if starting with i zero exponents. }

procedure EVINV (U,i,k: LIST): LIST;
{Exponent vector introduction of new variables. At position i in U k new variables are introduced. }

11.5 MAS Non-commutative Groebner Bases

procedure DINLNF (T,P,S: LIST): LIST;
{Distributive non-commutative polynomial left normal form. P is a list of non zero polynomials in distributive rational representation in r variables. S is a distributive rational polynomial. R is a polynomial such that S is left reducible to R modulo P and R is in normalform with respect to P. T is a table of distributive polynomials specifying the non-commutative relations. }

procedure DINLIS (T,P: LIST): LIST;
{Distributive non-commutative polynomial list left irreducible set. P is a list of distributive rational polynomials, PP is the result of left reducing each p element of P modulo P-(p) until

no further reductions are possible. T is a table of distributive polynomials specifying the non-commutative relations. }

procedure DINLSP (T,A,B: LIST): LIST;
{Distributive non-commutative polynomial left S-polynomial. A and B are rational polynomials in distributive representation. C is the left S-polynomial of A and B. T is a table of distributive polynomials specifying the non-commutative relations. }

procedure DINLGB (T,P,TF: LIST): LIST;
{Distributive non-commutative polynomials left Groebner base. P is a list of rational polynomials in distributive representation in r variables. PP is the left Groebner base of P. t is the trace flag. T is a table of distributive polynomials specifying the non-commutative relations. }

procedure DINLGM (T,P: LIST): LIST;
{Distributive non-commutative minimal ordered left Groebner base. P is a list of non zero rational polynomials in distributive representation in r variables, P is a left Groebner base. PP is the minimal normed and ordered left Groebner base. T is a table of distributive polynomials specifying the non-commutative relations. }

procedure DIN1GB (T,P,TF: LIST): LIST;
{Distributive non-commutative polynomials Groebner base. P is a list of rational polynomials in distributive representation in r variables. PP is the Groebner base of P. t is the trace flag. T is a table of distributive polynomials specifying the non-commutative relations. }

procedure DINCGB (T,P,TF: LIST): LIST;
{Distributive non-commutative polynomials Groebner base. P is a list of rational polynomials in distributive representation in r variables. PP is the Groebner base of P. t is the trace flag. T is a table of distributive polynomials specifying the non-commutative relations. }

Chapter 12

Arbitrary Domain Algorithms

12.1 Arbitrary Domain Tools

procedure ADRFFADIP (adip:LIST):LIST;
{arbitrary domain rational function from arbitrary domain intgeral polynomial. adip is an arbitrary domain integral polynomial. adip is converted to an arbitrary domain rational function. }

procedure ADDDFSTR (s:ARRAY OF CHAR):LIST;
{arbitrary domain domain descriptor from string. s is an string. The string must finish with a blank character. }

procedure ADCAST (e,dd:LIST):LIST;
{arbitraray domain cast. e is an element, dd is an arbitrary domain descriptor. The domain information of the elemnt e is changed to the domain informations of dd. No conversion is done. }

procedure ADRMDD (e: LIST):LIST;
{arbitrary domain remove domain descriptor informations. e is an arbitrary domain element. The domain descriptor informations are removed from the element e, the result is returned. }

procedure AdLoadConvFunc ();
{arbitrary domain load conversion functions. This procedure is called from the module MASLOADE after all domains are loaded. }

procedure RFDDFIPDD (ipdd:LIST):LIST;
{rational function domain descriptor from integral polynomial domain descriptor. }

procedure IPDDCMP (vlist:LIST):LIST;
{integral polynomial domain descriptor composition. }

procedure IPDECMP (e,vlist:LIST):LIST;
{integral polynomial domain element composition. }

procedure INTDDCMP ():LIST;
{integer domain descriptor composition. }

procedure IPDDADV (p: LIST; VAR q,r,vl:LIST);
{integral polynomial domain descriptor advance }

procedure RFDDADV (e: LIST; VAR rat, vl: LIST);
{rational function domain descriptor advance. }

12.2 Comprehensive-Groebner-Bases Applications

procedure GTEST (C,P: LIST; VAR C0,C1: LIST);
{Groebner-Test. C is a condition and P is a list of polynomials. PAR is a parameter for the factorization of coefficients. GTEST verifies, whether P is a comprehensive-groebner-basis wrt C. C0 contains the conditions so that P is no groebner basis. C1 contains the conditions so that P is a groebner basis. }

procedure GBHELP (COND,PPAIRS,P: LIST; VAR C0,C1: LIST);
{Groebner test help. COND is a condition. PPAIRS is the polynomial pairs list of P. P is a list of determined and coloured polynomials relative to COND. PAR is a parameter for the factorization of coefficients. C0 contains the successors of COND so that p is no groebner basis. C1 contains the successors of COND so that p is a groebner basis. }

procedure NSET (NN0,NN1,CC0,CC1: LIST; VAR C0,C1: LIST);
{Normalform set. NN0 is a set of tripels (ga,pco,c), where ga is a condition, pco is a normalform determined and coloured completely green by ga and c is a multiplicative factor. NN1 is a set of tripels (ga,pco,c), where ga is a condition, pco is a normalform determined and not coloured completely green by ga and c is a multiplicative factor. CC0 and CC1 are lists of conditions. The conditions in NN0, that are not in CC1, are added to CC0. The conditions in NN1 are added to CC1. }

procedure WRTEST (C,PP,CGB0,CGB1: LIST);
{Write groebner test. C is a condition. PP is a list of polynomials. CGB0 contains the successors of C so that p is no groebner basis. CGB1 contains the successors of C so that p is a groebner basis. The conditions are written on the output stream. }

procedure CPART (COND,CONDS: LIST): LIST;
{Condition part. COND is a condition. CONDS is a list of conditions. SL=1 if COND is a member of CNDS or a successor of a condition in CONDS. SL=0 else. }

procedure MCOEF (PCO: LIST; VAR COEFLI,COEF,TL: LIST);
{Make coefficient list. PCO is a coloured polynomial. COEFLI is the list of red and white coefficients in PCO, so that the degree of the terms in PCO is not zero. COEF=() if exists no monomial in PCO whose coefficient is coloured white and whose term is of deg zero or exists a monomial in PCO whose coefficient is coloured green and whose term has degree zero. COEF=0 if exists a monomial in PCO whose coefficient is coloured red and whose term is of deg zero. Else coef is the white coefficient in PCO whose term is of deg zero. TL = 1 if exists a red coloured coefficient whose term has degree ¿ 0. Else TL = 0 }

procedure NFEXEC (C,PPS,PP: LIST; VAR NOUT: LIST);
{Parametric ideal membership exec. C is a condition. PPS is a list of polynomials to be tested. PP is a list of polynomials forming a comprehensive-groebner-basis. PAR is a parameter for the factorization of coefficients. NOUT is a list containing each polynomial of PPS and the conditions and normalforms so that it is a member of the ideal and those so that it is no member of the ideal. }

procedure WRQFN0 (N0: LIST);
{Write quantifier free formula for parametric ideal membership. N0 is a list of tripel (cond,pco,c), where cond is a condition, pco is a normalform coloured completely green by cond and c is a multiplicative factor. The formula is written on the output stream. }

procedure DIMEXE (GS,V: LIST): LIST;
{Parametric dimension exec. GS is a groebner-system. V is the variables list. For each element of the groebner-system the dimension is computed. DIML is a list of conditions and dimensions. }

procedure INTDIM (V,PP: LIST): LIST;
{See intdim of dip. V is the variables list. PP is a list of polynomials. DL is the dimension of

PP. }

procedure WRDIMS (DIML: LIST);
{Write dimension. DIML is a list of conditions and dimensions. The quantifier free formula for parametric dimension is written on the output stream. }

procedure WRCONJ (CONDS: LIST);
{Write conjunction. CONDS is a list of conditions. The conjunction over all conditions is written on the output stream. }

12.3 Comprehensive-Groebner-Bases Data-Structures

procedure CondZero (Cond: LIST): LIST;
{Condition zero part. Cond is a condition. Returns the list of coefficients =0 in Cond.}

procedure CondNzero (Cond: LIST): LIST;
{Condition non-zero part. Cond is a condition. Returns the list of coefficients ¡¿0 in Cond.}

procedure CondParts (Cond: LIST; VAR C0, C1:LIST);
{Condition parts. Cond is a condition. C0,C1 need not be initialized. Returns the list of coefficients =0 from Cond in C0 and the list of coefficients ¡¿0 from Cond in C1 }

procedure CondCons (C0, C1:LIST):LIST;
{Condition construct. C0, C1 are lists of coefficients. Returns a condition where the coefficients from C0 are =0 and the coefficients from C1 are ¡¿0. }

procedure CondIsEmpty (Cond: LIST): BOOLEAN;
{Condition is empty. Cond is a condition. Returns true iff Cond is the empty condition. }

procedure CondEmpty ():LIST;
{Condition empty. Returns the empty condition. }

procedure CondRead (VD: LIST): LIST;
{Condition read. VD is a list containing a variable list and a domain descriptor. Returns a condition read from the input stream. }

procedure CondPRead (VD, B: LIST): LIST;
{Condition part read. VD is a list containing a variable list and a domain descriptor. Returns a condition part read from the input stream. }

procedure CondWrite (Cond: LIST);
{Condition write. Cond is a condition, which is written to the output stream. }

procedure FormFCond (Cond: LIST; VAR VD: LIST): LIST;
{Formula from Condition. Cond is a condition. DOM and VARL need not be initialized. Returns a formula "equivalent" to Cond and its variable list and domain descriptor in VD. }

procedure ColRed (Col: LIST): LIST;
{Colouring red. Col is a Colouring. Returns a list of red terms in Col. }

procedure ColWhite (Col: LIST): LIST;
{Colouring white. Col is a Colouring. Returns a list of white terms with white factors in Col. }

procedure ColParts (Col: LIST; VAR R, W:LIST);
{Colouring parts. Col is a Colouring. R, W need not be initialized. Returns a list of red terms from Col in R and a list of white terms with white factors in W. }

procedure ColCons (R, W: LIST): LIST;
{Colouring construct. R is a (possibly empty) list of red terms. W is a (possibly empty) list of white terms. Returns a colouring constructed from R an W. }

procedure ColConsCond (POL,COND: LIST): LIST;
{Colouring construct from condition. POL is a polynomial. COND is a condition. Returns a

colouring of POL according to COND. }

procedure ColIsEmpty (Col: LIST): BOOLEAN;
{Colouring is empty. Col is a Colouring. Returns true iff Col is the empty Colouring. }

procedure ColEmpty (): LIST;
{Colouring empty. Returns the empty Colouring. }

procedure ColH T (COL: LIST): LIST;
{Colouring head term. COL is a colouring. If the highest non-zero (red) term is greater then the highest unknown (white) term returns the highest non-zero term else returns SIL. }

procedure ColpCon s (POL,COL: LIST): LIST;
{Coloured polynomial construct. POL is a polynomial. COL is a colouring. Returns a Colp built from POL and COL.}

procedure ColpConsCond (POL,COND: LIST): LIST;
{Coloured polynomial construct from condition. POL is a polynomial. COND is a condition. Returns a Colp by colouring POL according to COND. }

procedure ColpPol (COLP: LIST): LIST;
{Coloured polynomial polynomial part. COLP is a coloured polynomial. Returns the polynomial in Colp. }

procedure ColpCol (COLP: LIST): LIST;
{Coloured polynomial colouring part. COLP is a coloured polynomial. Returns the colouring of COLP. }

procedure ColpParts (COLP: LIST; VAR POL,COL: LIST);
{Coloured polynomial parts. COLP is a coloured polynomial. POL, COL need not be initialized. Returns the polynomial from COLP in POL and the colouring from COLP in COL.}

procedure ColpIsCnst (COLP: LIST): BOOLEAN;
{Coloured polynomial is (non zero) constant. COLP is a coloured polynomial Returns TRUE iff COLP is constant (i.e. its degree is 0) and non zero wrt. to its colouring }

procedure ColpH T (COLP: LIST): LIST;
{Coloured polynomial head term. COLP is a coloured polynomial. If the head term of COLP is determined by its colouring (i.e. the highest non-zero (red) term is greater then the highest unknown (white) term) returns the head term else returns SIL. }

procedure ColpIsZer o (COLP: LIST): BOOLEAN;
{Coloured polynomial is zero. COLP is a coloured polynomial. Returns TRUE iff all terms of COLP are zero (green) wrt. its colouring. }

procedure CdRead (VD: LIST): LIST;
{Case distinction read. VD is a variable list and a domain descriptor. Returns a case distinction read from the input stream. }

procedure CdWrite (CD: LIST);
{Case distinction write. CD is a case distinction which is written to the output stream. }

procedure CdpCons (CD,PL,VD: LIST): LIST;
{Case distinction and polynomial set construct. CD is a case distinction. PL is a list of Polynomials. VD is a varable list and domain descriptor for PL. Returns a CDP built from CD,PL,VD.}

procedure CdpParts (CDP: LIST; VAR CD,PL,VD: LIST);
{Case distinction and polynomial set parts. CDP is a case distinction and polynomial set. Returns the case distinction from CDP in CD, the polynomial set from CDP in PL and the variable list and domain descriptor from CDP in VD. }

procedure CdpCd (CDP: LIST): LIST;
{Case distinction and polynomial set case distinction part. CDP is a case distinction and polynomial set. Returns the case distinction from CDP. }

procedure CdpPs (CDP: LIST): LIST;
{Case distinction and polynomial set polynomial set part. CDP is a case distinction and polynomial set. Returns the polynomial set from CDP. }

procedure CdpVd (CDP: LIST): LIST;
{Case distinction and polynomial set variable list and domain descriptor part. CDP is a case distinction and polynomial set. Returns the variable list and domain descriptor from CDP. }

procedure CdpRead ():LIST;
{Case distinction and polynomial set read. Returns a case distinction and polynomial set read from the input stream.}

procedure CdpWrite (CDP: LIST);
{Case distinction and polynomial set write. CDP is a case distinction and polynomial set. Writes CDP to the output-stream.}

procedure RDSYS (VD: LIST): LIST;
{Read polynomial systems. VD is a variable list and domain descriptor. PPS is a list of two polynomial systems read from the input stream. }

procedure GsCons (S,VD,CD: LIST): LIST;
{Groebner system construct. S is a Groebner system, VD its variables list and domain descriptor and CD the initial case distinction Returns the Groebner system constructed from S, VD and CD }

procedure GsS (GS: LIST): LIST;
{Groebner system system part. GS is a Groebner system. Returns the system part of GS }

procedure GsVd (GS: LIST): LIST;
{Groebner system variable list and domain descriptor. GS is a Groebner system. Returns the variable list and domain descriptor for GS. }

procedure GsCd (GS: LIST): LIST;
{Groebner system initial case distinction. GS is a Groebner system. Returns the initial case distinction for GS. }

procedure GsParts (GS: LIST; VAR S,VD,CD: LIST);
{Groebner system parts. S and VD need not be initialized. Returns the system part of GS in S, the variable list and domain descriptor for GS in VD and the initial case distinction in CD. }

procedure GsWrite (GS: LIST);
{Groebner system write. GS is a Groebner system which is written to the output stream. }

procedure CgbCons (CGB,I,VD,CD: LIST): LIST;
{Groebner system construct. CGB is a list of polynomials, I is an Integer, the number of conditions, VD its variables list and domain descriptor and CD the initial case distinction Returns the comprehensive Groebner basis constructed from CGB, I, VD and CD }

procedure CgbP (CGB: LIST): LIST;
{Comprehensive Groebner basis polynomial list part. CGB is a comprehensive Groebner basis Returns the polynomial list part of CGB }

procedure CgbI (CGB: LIST): LIST;
{Comprehensive Groebner basis number of conditions part. CGB is a comprehensive Groebner basis Returns the number of conditions part of CGB }

procedure CgbVd (CGB: LIST): LIST;
{Comprehensive Groebner basis variable list and domain descriptor. CGB is a Comprehensive Groebner basis. Returns the variable list and domain descriptor for CGB. }

procedure CgbCd (CGB: LIST): LIST;
{Groebner system initial case distinction. CGB is a Comprehensive Groebner basis. Returns the initial case distinction for CGB. }

procedure CgbParts (CGB: LIST; VAR P,I,VD,CD: LIST);
{Comprehensive Groebner basis parts. P, I, VD and CD need not be initialized. Returns the polynomial list part of CGB in P, the number of conditions in I, the variable list and domain descriptor for CGB in VD and the initial case distinction in CD. }

procedure CgbWrite (CGB: LIST);
{Comprehensive Groebner basis write. CGB is a comprehensive Groebner basis which is written to the output stream }

procedure FdCons (F,D,V: LIST): LIST;
{Formula and dimension construct. F is a formula. D is an integer. Returns the Fd constructed from F and D. }

procedure FdParts (FD: LIST; VAR F,D,V: LIST);
{Formula and dimension parts. FD is a formula and dimension. F, D and V need not be initialized. Returns the formula from FD in F, the dimension in D and the variable list in V. }

procedure FdF (FD: LIST): LIST;
{Formula and dimension formula part. FD is a formula and dimension. Returns the formula part from FD}

procedure FdD (FD: LIST): LIST;
{Formula and dimension formula part. FD is a formula and dimension. Returns the dimension part from FD}

procedure FdV (FD: LIST): LIST;
{Formula and dimension variable list part. FD is a formula and dimension. Returns the variable list part from FD}

procedure FdWrite (FD: LIST);
{Formula and dimension write. FD is a formula and dimension, which is written to the output stream. PQPRING must be called prior to calling FdWrite to set domain descriptor and variable list. }

procedure PdCons (F,VD: LIST): LIST;
{Parametric dimension construct. F is a formula. VD is the variable list and domain descriptor for F. Returns the Pd constructed from F and V. }

procedure PdParts (PD: LIST; VAR F,VD: LIST);
{Parametric dimension parts. Pd is a parametric dimension. F and V need not be initialized. Returns the formula from PD in F and the variable list and domain descriptor in V. }

procedure PdF (PD: LIST): LIST;
{Parametric dimension formula and dimension list part. PD is a parametric dimension. Returns the formula and dimension list from PD. }

procedure PdVd (PD: LIST): LIST;
{Parametric dimension variable list and domain descriptor part. PD is a Parametric dimension. Returns the dimension from PD. }

procedure PdWrite (PD: LIST);
{Parametric dimension write. PD is a Parametric dimension, which is written to the output stream. }

procedure VdCons (V,D: LIST): LIST;
{Variable list and domain descriptor construct. V is a variable list. D is a domain descriptor. Returns the VD constructed from V and D. }

procedure VdV (Vd: LIST): LIST;
{Variable list and domain descriptor variable list part. Vd is a variable list and domain descriptor. Returns the variable list from Vd. }

procedure VdD (Vd: LIST): LIST;
{Variable list and domain descriptor domain descriptor part. Vd is a variable list and domain descriptor. Returns the domain descriptor from Vd. }

procedure VdParts (Vd: LIST; VAR V,D: LIST);
{Variable list and domain descriptor parts. Vd is a variable list and domain descriptor. Returns the variable list from Vd in V and the domain descriptor in D. }

procedure VdRead (): LIST;
{Variable list and domain descriptor read. Returns a variable list and domain descriptor read from the input stream.}

12.4 Comprehensive-Groebner-Bases Utility Functions

procedure WRCOL (COL,POL: LIST);
{Write colour. POL is a polynomial. COL contains the red and the white coloured terms of POL. The red and white monomials of POL are written on the output stream. }

procedure WRTERM (TERM,V: LIST);
{Write term. TERM is a term. V is the variable list. Term is written on the output stream. }

procedure DWRIT (DE: LIST);
{Distinction write. DE is a case distinction. DE is written on the output stream. }

procedure CGBCOL (COND,PL: LIST): LIST;
{Write coloured polynomials without green monomials. COND is a condition. PL is a list of polynomials coloured wrt the condition. If the condition contains coefficients to be 0, each polynomial is written on the output stream without the green coloured monomials. }

procedure DCLWR (PL,B: LIST);
{Coloured polynomials list write. PL is a list of coloured polynomials. If B = 0 the polynomial list is written on the output stream. If B = 1 the polynomials and the red and white monomials are written on the output stream. }

procedure FINDCP (TTERM,WHITE: LIST): LIST;
{Find white factors. TTERM is a term, WHITE is a list of pairs, containing a white coloured term and his list of white coloured factors of the coefficient. If white contains tterm, CP is the list of TTERM and the white factors. else CP is empty. }

procedure FINDBC (RE,POL: LIST): LIST;
{Find base coefficient. RE is a term. POL is a polynomial, where RE is one of the terms of POL. PA is the base coefficient of RE in POL. }

procedure FINDRM (RE,POL: LIST): LIST;
{Find monomial. RE is a term. POL is a polynomial, where RE is one of the terms of POL. RPOL is the polynonomial, containing only the monomial with RE in POL. }

procedure CGBFRM (CGBL: LIST): LIST;
{Comprehensive-Groebner-Basis from coloured basis. CGBL is a list of coloured polynomials. CGB is a list of the polynomials in CGBL (without colours). }

procedure MKPOL (PCO: LIST): LIST;
{Make polynomial without green monomials. PCO is a coloured polynomial. PPOL is the polynomial without green monomials. If the polynomial is completely coloured green, PPOL is empty. }

procedure GREPOL (PL: LIST): LIST;
{Get polynomials without green monomials. PL is a list of coloured polynomials. X is a list of the polynomials in PL without green monomials. }

procedure WMEMB (TTERM,WHITE: LIST): LIST;
{White term member. TTERM is a term, white is a list of pairs, containing a white coloured term and his list of white coloured factors of the coefficient. SL=1 if white contains TTERM, else SL=0. }

procedure EQPLCL (ALIST,BLIST: LIST): LIST;
{Equal lists of coloured polynomials. ALIST and BLIST are lists of coloured polynomials. SL = 1 if the polynomials in ALIST and BLIST are the same. Else SL = 0. }

procedure CGBLM (L1,L2: LIST): LIST;
{CGB coloured distributive polynomial list merge. L1 and L2 are lists of coloured distributive polynomials in non decreasing order. The merger of L1 and L2 is returned. (This procedure is a modified version of DIPLM from DIPC.MOD which does the same for - not coloured - distributive polynomials.)}

procedure CGBLPM (A: LIST): LIST;
{CGB list merge. A is a list of couloured polynomials. B contains the coloured polynomials in a in nondecreasing order wrt to their colour. See DIPLPM. }

procedure ADDCON (COEFL,COND: LIST): LIST;
{Add to condition. COEFL is a list of coefficients. COND is a condition. Returns a case distinction covering COND containing all possible cases for COEFL }

procedure INICOL (COND,PI: LIST): LIST;
{Initialize colour. COND is a condition. PI is a polynomial. COL is the list of red terms and white terms (with white part) of PI wrt to the condition. }

procedure SETCOL (COND,COL: LIST): LIST;
{Set colour. COND is a condition. COL is a list of red terms and white terms (with white part) wrt another condition, such that COND is a successor of this condition. COL is updated to COLS wrt COND. }

procedure REDSRT (RALT,RNEU: LIST): LIST;
{Red terms sort. RNEU and RALT are lists of terms in nondecreasing order. CRED0 contains the terms of RALT and RNEU in nondecreasing order. }

procedure TESTHT (COL: LIST):LIST;
{Test highest term. COL contains a list of red terms and a list of white terms. CP contains the highest white term and its white part if it is gt the highest red term. Else CP is empty. }

procedure DETPOL (GA,PI,COL: LIST; VAR DLIST,CLIST: LIST);
{Determine polynomial. GA is a condition. PI is a polynomial. COL contains the list of the red terms of PI and the list of the white terms of PI wrt the condition. DLIST is a case distinction that covers GA and determines PI. CLIST is a list of pairs each containing a condition of DLIST and PI coloured wrt this condition. }

procedure DET (CONDS,P: LIST; VAR DLIST,PPL: LIST);
{Determine list of polynomials. CONDS is a case distinction. P is a list of polynomials. DLIST is a case distinction that covers CONDS and determines P. PPL is a list of pairs each containing a condiotion of DLIST and P coloured wrt this condition. }

procedure VERIFY (D,CLIST: LIST): LIST;
{Verify conditions and polynomials. D is a case distinction with the conditions c1,... ,cn. CLIST is a list of pairs each containg a condition and a coloured polynomial. The structure of clist is ((c11,p1),... ,(cn1,p1), (c12,p2),... ,(cn2,p2), (c1m,pm),... (cnm,pm)). p1,... ,pm are coloured polynomials wrt cij. ci is a successor of cij and cim=ci for i=1; n, j=1; m. The structure of the result is q((c1,(p11,... ,p1m)),... , (c2,(p21,... ,p2m)),... , (cn,(pn1,... ,pnm))), where (pi1,... ,pim) is a permutation of (p1,... ,pm), so that the polynomials are in nondecreasing order wrt the condition pi for i=1; n. }

procedure AINB (ALIST,BLIST: LIST): LIST;
{A in B. ALIST and BLIST are lists of coefficients. SL eq 1 if all elements of ALIST are in BLIST. Else SL eq 0. }

12.5 Comprehensive-Groebner-Bases Main Programms

procedure CDINIT (CD: LIST): LIST;
{Case distinction init. CD is a case distinction. Returns a case distinction with conditions as required by PAR.Cond*. }

procedure GSYS (CDP: LIST): LIST;
{Groebner system. CDP is case distinction and polynomial set. Returns a Groebner system for CDP. }

procedure GSYSF (CDP: LIST): LIST;
{Groebner system with factorization. CDP is case distinction and polynomial set. Returns a Groebner system for CDP. }

procedure GSYSDIM (GS: LIST): LIST;
{Groebner system dimension. GS is a Groebner system. Returns the parametric dimension for GS. }

procedure DIMIS (PL,VL: LIST; VAR MAXVL: LIST): LIST;
{Dimension and maximal independent set. PL is a list of polynomials. VL is the variable list. MAXVL need not be initialized. Returns the dimension of PP and a maximal independent set in MAXVL. }

procedure GSYSRED (GS: LIST): LIST;
{Reduce Groebner system. GS is a Groebner system. Returns a reduced Groeber system for GS. }

procedure CGBFGSYS (S: LIST): LIST;
{Comprehensive Groebner basis from Groebner system. S is a Groebner system. Returns a comprehensive Groebner basis. }

procedure CGBGLOBRED (CGB: LIST): LIST;
{Comprehensive Groebner basis global reduce. CGB is a comprehensive Groebner basis. Returns a global reduced comprehensive Groebner basis. }

procedure CGBQFF (CGB: LIST): LIST;
{Comprehensive Groebner basis quantifier free formula. CGB is a comprehensive Groebner basis. Returns a formula containing a condition for the existence of common zeroes of the polynomials in CGB. }

procedure CGBIN ();
{Comprehensive-Groebner-Basis input. The input is read from the stream. Start computation by call of CGBOUT. }

procedure CGBOUT (AC: LIST);
{Comprehensive-Groebner-Basis execute and output. AC contains the input data set (case distinction, 2 polynomial systems, polynomial descriptor, list of options). }

procedure DVREAD (): LIST;
{Polynom descriptor read. }

procedure CONINI (VD: LIST): LIST;
{Initialize case distinction. VD is the domain descriptor. CONS is the case distinction read from the input stream. }

procedure CONDRD (V,D,B,DALT: LIST; VAR DNEU: LIST);
{Conditions read. V is the variables list, D is the domain descriptor, DALT is a case distinction. DNEU contains DALT and new coefficients, which are zero, if B=0. If B=1 they are not zero. }

procedure UPDCAS (ALIST,DALT,B: LIST): LIST;
{Update case distinction. ALIST is a list of coefficients (a1,... ,an). DALT is a case distinction. If B=0 then DNEU is a case distinction including DALT and (a1=0,... , an=0). If B=1 then DNEU is a case distinction including DALT and (a1¡¿0,... , an¡¿0). ADDCON computes a complete case distinction including DALT and (a1,... ,an). Then the well formed conditions are composed. }

procedure CCOVER (CONS: LIST): LIST;
{Cover condition. CONS is a case distinction. C is a condition, so that CONS covers C. }

procedure SCOV (CONDA,CONS,B: LIST): LIST;
{Search condition. CONDA is a list of coefficients. CONS is a list of conditions. If B=0 then SCOV returns all coefficients, that are in CONDA and in the zero list of each condition in CONS. If B=1 then SCOV returns all coefficients, that are in CONDA and in the not-zero list of each condition in CONS. }

procedure CHDOM (CONDS,PPS: LIST; VAR CONS,PP: LIST);
{Change domain. CONDS is a case distinction. PPS is a list of polynomials with coefficient from an arbitrary domain. This list is converted to a list PP of integral polynomials. Each polynomial containing fractions, is mutliplied with the lcm of the coefficient- denominators. CONS contains CONDS and conditions to assure that the prime-factors of each lcm are not zero. This procedure makes sense for rational-polynomials only. For integral-polynomials it will work, but create overhead by copying PPS to PP }

procedure EXECRD (): LIST;
{Exec read. The list nrlist of options is read from the input stream. }

procedure SEENR (AC: LIST; VAR NR: LIST);
{Find key for option. AC is an option. NR is the key for AC. }

procedure WRTITL (NR: LIST);
{Write title. }

procedure WRGBS (PLS: LIST);
{Write groebner-system. PLS is a list of pairs, each pair containing a condition and a polynomials list, where each polynomial is coloured wrt the condition. The conditions and the polynomials are written on the output stream, sorted by polynomial systems. }

procedure WRCGB (CGB,I: LIST);
{Write comprehensive-groebner-basis. CGB is a list of coloured polynomials forming a comprehensive-groebner-basis. I is the number of conditions of the groebner-system. CGB and I are written on the output stream. }

procedure WRRCGB (CGB,I: LIST);
{Write reduced comprehensive-groebner-basis. CGB is a list of coloured polynomials forming a reduced comprehensive-groebner-basis. I is the number of conditions of the groebner-system. CGB and I are written on the output stream. }

procedure GGREEN (GS: LIST);
{Write groebner-system without green monomials. GS is a list of pairs, each pair containing a condition and a polynomials list, where each polynomial is coloured wrt the condition. The conditions and the polynomials are written on the output stream without green coloured monomials. }

procedure NWRIT0 (N: LIST);
{Normalforms write. N is a set of tripels each containing a condition, a polynomial coloured green wrt the condition and a factor c. The polynomials form a set of normalforms. The conditions and the factors are written on the output stream. }

procedure NWRIT1 (N: LIST);
{Normalforms write. N is a set of tripels each containing a condition, a polynomial not coloured green wrt the condition and a factor c. The polynomials form a set of normalforms. The conditions, the polynomials and the factors are written on the output stream. }

procedure WPAIRS (PS: LIST);
{Write pairs of polynomials. PS is a list of tripels, each tripel containing two coloured polynomials and the product of their highest terms wrt their colour. The polynomials are written on the output stream. }

procedure WPLIST (PL: LIST);
{Write polynomials and pairs. PL is a list of tripels, each tripel containing a condition, a set of polynomials coloured wrt the condition and a set of pairs of polynomials, constructed from the set of polynomials. The condition, the polynomials and the pairs are written on the output stream. }

procedure NFWRIT (NOUT: LIST);
{Normalforms write. The polynomials for which the test for parametric ideal membership has been executed are written on the output stream their normalforms. For each polynomial the quantifier free formula is written on the output stream. }

12.6 Comprehensive-Groebner-Bases Miscellaneous Programs

procedure EvordSet (T: LIST);
{EVORD set. T is a termorder. The global variable EVORD is set to T. The old value of EVORD is put on top of a stack and can be restored using EvordReset(). }

procedure EvordReset ();
{Reset evord. The global variable EVORD is set to the top element of EVORDSTACK. (EVORDSTACK is set by EvordSet().) }

procedure ValisSet (V: LIST);
{Set valis. V is a variables list. The global variable VALIS is set to T. The old value of VALIS is put on top of a stack and can be restored using ValisReset(). }

procedure ValisReset ();
{Reset valis. The global variable VALIS is set to the top element of VALISSTACK. (VALISSTACK is set by ValisSet().) }

procedure SetInsert (e, A: LIST): LIST;
{Set insert. A is a set. e is an element. Returns the set A U e.}

procedure SetUnion (A,B: LIST): LIST;
{Set union. A is a set. B is a set. Returns the set A U B. }

procedure CGBOPT (O: LIST);
{Comprehensive Groebner Basis Options. O is a list with an arbitrary number of elements. The global variable PAR is set according to O. The elements of O (if existent) are interpreted as follows: 1st element: if =0 no output during computation, if ¿0 chatty. 2nd element: if =1 factorize coefficients, if ¡¿ do not. 3rd element: if =0 use top reduction only, if =1 use "normal" reduction. 4th element: evaluate conditions using: if =0: simple methode, if =1: reduced sets, if =2: Groebner bases. 5th element: if =0: characteristic 0, if ¡¿0 arbritrary characteristic. 6th element: term order for polynomials 7th element: term order for coefficients }

procedure CGBOPTWRITE ();
{Comprehensive Groebner Basis Options Write Writes the options from the global Variable PAR on the output stream}

procedure dummycnst (A: LIST): BOOLEAN;
{Dummy constant. Value for PAR.IsCnst. Returns false always (nothing is constant). }

procedure dummyfactorize (A: LIST): LIST;
{Dummy factorize. Value for PAR.factorize. Does not factorize. Returns a list containing A.}

procedure FLWRITE (L: LIST);
{Formatted list write. The input list L is written to the output stream.}

procedure FILWRITE (L: LIST; N:INTEGER);
{Formatted indented list write. The input list L is written to the output stream.}

procedure XPFDI P (DP, DOM, VARL: LIST): LIST;
{Recursive polynomial (with domain-descriptor) from distributive polynomial. DP is a polynomial in distributive representation. DOM is a domain descriptor. VARL is a list of variables. Returns a Polynomial (DOM, P, R, VARL) where P is the recursive representation of DP and R is the number of variables of DP. }

procedure PFLDIP L (DIPL, DOM, VARL: LIST): LIST;
{Recursive polynomial list (with domain-descriptor) from distributive polynomial list. DIPL is a list of polynomials in distributive representation. DOM is a domain descriptor. VARL is a list of variables. Returns a list containing an element (DOM, P, R, VARL) for each distributive polynomial dp in DIPL where P is the recursive representation of dp and R is the number of variables of dp (all polynomials in DIPL are assumed to have the same number of variables). }

procedure XDIPFP F (P: LIST; VAR DOM, VARL: LIST): LIST;
{Distributive polynomial from recursive polynomial (with domain-descriptor). P is a polynomial in recursive representation. Returns this polynomial in distributive representation, sorted according to the value of EVORD, the domain-descriptor in DOM and the list of variables in VARL. }

procedure DIPLFPF L (PFL: LIST; VAR DOM, VARL: LIST): LIST;
{Distributive polynomial list from recursive polynomial (with domain-descriptor) list. PFL is a list of polynomials in recursive representation. Returns a list of this polynomials in distributive representation the domain-descriptor in DOM and the list of variables in VARL. }

procedure DIFPF (P, D: LIST; VAR DOM, VARL: LIST): LIST;
{Distributive polynomial with arbitrary domain coefficients from recursive polynomial (with domain-descriptor). P is a polynomial with domain descriptor. D is a domain descriptor. Returns P in distributive representation over domain D, sorted according to the value of EVORD, the domain-descriptor of P in DOM, and the list of variables in VARL. }

procedure DILFPFL (PFL, D: LIST; VAR DOM, VARL: LIST): LIST;
{Distributive polynomial list with arbitrary domain coefficients from recursive polynomial list (with domain-descriptor). P is a polynomial list with domain descriptor. D is a domain descriptor. Returns a list containing the polynomials from PFL in distributive representation over domain D, sorted according to the value of EVORD, the domain-descriptor of PFL in DOM, and the list of variables in VARL. }

procedure PFIGB (PFL, TF: LIST; VAR ONE: BOOLEAN): LIST;
{Integral Polynomial Groebner Basis. PFL is a list of polynomials in recursive representation. TF is the trace flag. Returns the Groebner Basis of PFL wrt. to the total degree inverse lexicographical term order. ONE=TRUE iff 1 is an element of the Groebner Basis.}

procedure PFIGBA (PFL, P, TF: LIST; VAR ONE: BOOLEAN): LIST;
{Integral Polynomial Groebner Basis augmentation. PFL is a list of polynomials in recursive representation. P is a polynomial. TF is the trace flag. Returns the Groebner Basis of PFL and P wrt. to the total degree inverse lexicographical term order. ONE=TRUE iff 1 is an element of the Groebner Basis.}

procedure PFILS (B: LIST; VAR ONE: BOOLEAN): LIST;
{Integral polynomial list irreducible set. B is a list of polynomials in recursive representation. Returns the result of reducing B. ONE=TRUE iff 1 is an element of the result. }

procedure DIILIS (P: LIST): LIST;
{Distributive integral polynomial list irreducible set. P is a list of distributive integral polynomials, PP is the result of reducing each p element of P modulo P-(p) until no further reductions are possible. }

procedure PFINOR (B, P: LIST): LIST;
{Integral Polynomial Normal Form. B is a list of polynomials in recursive representation. P is a polynomial in recursive representation. Returns the normal form of P wrt. B, or SIL if this normal form is 0. }

procedure PFILNOR (B, P: LIST; VAR ZERO: BOOLEAN): LIST;
{Integral Polynomial List Normal Form. B is a list of polynomials in recursive representation. P is a list of polynomials in recursive representation. Returns a list of (non-zero, not constant) normal forms of each p in P wrt. B. ZERO=TRUE iff one of the normal forms is zero. }

procedure PFILDS (B: LIST; VAR ONE: BOOLEAN): LIST;
{Integral polynomial list d-irreducible set. B is a list of polynomials in recursive representation. Returns the result of d-reducing B. ONE=FALSE.}

procedure PFIDNOR (B, P: LIST): LIST;
{Integral Polynomial D Normal Form. B is a list of polynomials in recursive representation. P is a polynomial in recursive representation. Returns the d-normal form of P wrt. B, or SIL if this normal form is 0. }

procedure PFILDNOR (B, P: LIST; VAR ZERO: BOOLEAN): LIST;
{Integral Polynomial List D-Normal Form. B is a list of polynomials in recursive representation. P is a list of polynomials in recursive representation. Returns a list of (non-zero) d-normal forms of each p in P wrt. B. ZERO=TRUE iff one of the d-normal forms is zero. }

procedure PFWRITE (P: LIST);
{Integral polynomial write. P is a polynomial in recursive representation with domain-descriptor. P is written to the outputstream (wrt. the term order in EVORD). }

procedure DIIPNORM (P: LIST): LIST;
{Distributive integral polynomial norm. Returns a polynomial r, were n*r=p for an Integer n, the content of r is 1 and the highest coefficient of r is not negative.}

12.7 Comprehensive-Groebner-Bases System

procedure GRED (COND,PCO,PCI,RE: LIST; VAR RCO,HA: LIST);
{Parametric reduction. COND is a condition, PCO and PCI are coloured polynomials. PCI is determined by cond. RE is a term in PCO coloured red or white by COND. RE is a multiple of the headterm (wrt cond) of PCI. RCO is the result of one step reduction of PCO (by PCI) to eliminate the term RE. HA is the multiple factor of PCO. }

procedure GBDIFF (COND,A,ACOLS,B,BCOLS: LIST; VAR C,CCOLS: LIST);
{Parametric difference. COND is a condition. A and B are polynomials. ACOLS is the colouring of A wrt cond, BCOLS is the colouring of B wrt COND. C=A-B. CCOLS is the colouring of C wrt COND. }

procedure COLPRD (COL1,TTERM: LIST): LIST;
{Colour product. COL1 contains a list of red terms and a list of white terms. TTERM is a term. Every term in COL1 is multiplied with TTERM.}

procedure CMULT (ONECL,TTERM,B: LIST): LIST;
{Colour multiplication. If B=1 then ONECL is a list of (red) terms. If B=2 then ONECL is a list of pairs, each containing a (white) term and the white part of its coefficient. TTERM is a term. Every term in ONECL is multiplicated with TTERM. CCOL is the result. }

procedure WHSRT (COL,TTERM,ALIST: LIST): LIST;
{White sort. COL contains a list of red terms and a list of white terms. The form of COL is ((r1,... ,rn),((w1,(wp11,; ,wp1s)),... ,(wm,(wpm1,; ,wpms)))). TTERM is a term. ALIST is a list of coefficients. The form of ALIST is (a1,... ,at). Every term of COL is multiplied with TTERM. The resulting terms are coloured white by adding ALIST to its white part. The list of red terms is empty. The list of white terms z is as follows ((r1*tterm,(a1,... ,at)),... ,(rn*tterm,(a1,... ,at)), (w1*tterm,(a1,... ,at,wp11,... ,wp1s)),... , (wm*tterm,(a1,... ,at,wpm1,... ,wpms))). CWHIT0 contains the same terms as z in a nondecreasing order. COLS is pair containg an empty list of red terms and the list CWHIT0. }

procedure WUPD (ALIST,BLIST: LIST): LIST;
{White part update. ALIST and BLIST are sets of coefficients. Returns the union of ALIST and BLIST. }

procedure COLDIF (T,ACOLS,COLR,COLW: LIST; VAR CRED,CWHITE: LIST);
{Colour difference. T is term. ACOLS contains a list of red terms and a list of white terms. COLR is a list of red terms. COLW is a list of white terms. If T is member of the red terms in ACOLS, it is added to COLR. The result is CRED. If T is member of the white terms in ACOLS, it is added to COLW with its white part. The result is CWHITE. }

procedure KEYCOL (EL,ACOLS: LIST; VAR KEY,ALIST: LIST);
{Key colour. EL is a term. ACOLS contains a list of red terms and a list of white terms. If EL is member of the red terms in ACOLS then KEY=1 and ALIST is empty. If EL is member of the white terms in ACOLS then KEY=2 and ALIST is the white part of EL. If EL is not in ACOLS (EL is coloured green) then KEY=0 and ALIST is empty. }

procedure MKACOL (ALIST,EL,COLR,COLW: LIST; VAR CRED,CWHITE: LIST);
{Make colour. ALIST is a list of coefficients. EL is a term. COLR is a list of red terms. COLW is a list of white terms. If ALIST is empty, EL is added to COLR. The result is CRED. If ALIST is not empty, the pair of EL and ALIST is added to COLW. the result is CWHITE. }

procedure MKCOL (COND,CA,CE,COLR,COLW: LIST; VAR CRED,CWHITE: LIST);
{Make new colour. COND is a condition. CA is a coefficient. CE is a term. COLR is a list of red terms. COLW is a list of white terms. If CA is coloured red by COND, CE is added to COLR. the result is CRED. If CA is coloured white by COND, the pair with CE and the white factors of CA are added to COLW. the result is CWHITE. }

procedure FINCOL (APP,ACOLS,COLR,COLW: LIST; VAR CRED,CWHITE: LIST);
{Finish colouring. APP is a polynomial. ACOLS contains a list of red terms and a list of white terms. COLR is a list of red terms. COLW is a list of white terms. The red terms of APP are added to COLR. the result is CRED. The white terms of APP are added to COLW with their white part. The result is CWHITE. }

procedure NFORM (GA,FCO,P: LIST; VAR N0,N1: LIST);
{Parametric normalform. GA is a condition. FCO is a polynomial coloured wrt GA. P is a list of polynomials coloured wrt GA. FCO is reduced modulo P wrt GA. N0 is the set of tripel of the form (cond,pco,c), where cond is a condition, pco is a normalform of fco coloured completely green by cond and c is a multiplicative factor. N1 is the set of tripel of the form (cond,pco,c), where cond is a condition, pco is a normalform of fco not coloured completely green by cond and c is a multiplicative factor. }

procedure NFTOP (GA,FCO,P: LIST; VAR N0,N1: LIST);
{Normalform by topreduction. GA is a condition. FCO is a polynomial coloured wrt GA. P is a list of polynomials coloured wrt GA. FCO is reduced modulo P wrt GA. N0 is the set of tripel of

the form (cond,pco,c), where cond is a condition, pco is a normalform of fco coloured completely green by cond and c is a multiplicative factor. N1 is the set of tripel of the form (cond,pco,c), where cond is a condition, pco is a normalform of fco not coloured completely green by cond and c is a multiplicative factor. }

procedure FINDPI (PCO,P: LIST; VAR PCI,RE: LIST);
{Find polynomial. PCO is a coloured polynomial. P is a list of coloured polynomials. A polynomial for the reduction over all terms of PCO modulo P is searched. PCI is emtpy if no polynomial is found, else PCI is the found polynomial and RE is the term of PCO to be eliminated. }

procedure FINDPITOP (PCO,P: LIST; VAR PCI,RE: LIST);
{Find polynomial. PCO is a coloured polynomial. P is a list of coloured polynomials. A polynomial for the topreduction of PCO modulo P is searched. PCI is emtpy if no polynomial is found, else PCI is the found polynomial and RE is the term of PCO to be eliminated. }

procedure SPOL (COND,HA,HB: LIST): LIST;
{Parametric spolynom. COND is a condition. HA and HB are coloured polynomials. Returns the spolynom of HA and HB with colouring. }

procedure GBSYS (CNDS,P: LIST): LIST;
{Groebner system. CNDS is a case distinction. P is a list of polynomials. Returns a Groebner-system for P relative to CNDS. }

procedure GBSYSF (CNDS,P: LIST): LIST;
{Groebner system with factorization. CNDS is a case distinction. P is a list of polynomials. Returns a Groebner-system for P relative to CNDS. }

procedure VRNORM (COND,PP,N0,N1,PPAIRS: LIST;
VAR P,PAIRS,PAIRSL,GSYS: LIST); {Verify normalforms. COND is a condition. PP is a polynomials list determined by COND. N0 is a set of tripel (ga,pco,c), where ga is a condition, pco is a normalform determined and coloured completely green by ga and c is a multiplicative factor. N1 is a set of tripel (ga,pco,c), where ga is a condition, pco is a normalform determined and not coloured completely green by ga and c is a multiplicative factor. PPAIRS is the polynomials pairs list of PP. The normalforms are checked. PP is updated to P. PPAIRS is updated to PAIRS. PAIRSL is a list of the form (cond1,p1,pairs1,... ,condn,pn,pairsn) constructed from the information of the N0 and N1. GSYS is a list of pairs, each containing a condition and a groebner base wrt this condition. }

procedure CHDEGL (PLIST: LIST): LIST;
{Check degree of polynomial list. PLIST is a list of coloured polynomials. PCO is an element of PLIST, such that the degree wrt the colouring of the polynomial is 0. PCO is emtpy if no such polynomial exists. }

procedure MKN1 (NN1,P,PAIRS: LIST; VAR PPLIST,GSYS: LIST);
{Make n1. NN1 is a set of tripel of the form (ga,pco,c), where ga is a condition, pco is a normalform determined and not coloured completely green by ga and c is a multiplicative factor. P is a list of coloured polynomials. PAIRS is the polynomials pairs list of P. PPLIST is a list of the form (cond1,p1,pairs1,... ,condn,pn,pairsn) constructed from the information of the NN1. GSYS is a list of pairs, each containing a condition and a groebner basis wrt this condition. }

procedure MKN0 (NN0,P,PAIRS: LIST; VAR PPLIST: LIST);
{Make n0. NN0 is a set of tripel of the form (ga,pco,c), where ga is a condition, pco is a normalform determined and coloured completely green by ga and c is a multiplicative factor. P is a list of coloured polynomials. PAIRS is the polynomials pairs list of P. PPLIST is a list of the form (cond1,p1,pairs1,... ,condn,pn,pairsn) constructed from the information of the NN0. }

procedure GSYSN0 (NN0,P: LIST; VAR GSYS: LIST);
{Groebner-System n0 update. NN0 is a set of tripel of the form (ga,pco,c), where ga is a condition, pco is a normalform determined and coloured completely green by ga and c is a multiplicative factor. P is a list of coloured polynomials. for each GA in NN0, the pair of the form (ga,p) is

added to GSYS. }

procedure MINPP (PP,PCO: LIST): LIST;
{Minimize polynomials list. PP is a list of coloured polynomials. PCO is a coloured polynomial. P is a list of PCO and those polynomials in PP, such that their headterms wrt the colouring can not be divided by the headterm of PCO relative to its colouring. }

procedure UPDPP (COND,P: LIST): LIST;
{Update polynomials. COND is a condition. P is a list of polynomials determined and coloured wrt a predecessor of COND. The colouring of each polynomial is updated. The result is the polynomial-list PP. }

procedure GBUPD (COND,P,GBSYS: LIST): LIST;
{Groebner-system update. COND is a condition. P is a list of polynomials determined and coloured wrt a predecessor of COND. GBSYS is the actual Groebner-system. The colouring of each polynomial in P is updated relative to COND. Extraneous polynomials are elminiated. The condititon and the resulting polynomials list are added to GBSYS. The result is GSYS. }

procedure MKPAIR (PP: LIST): LIST;
{Make pairs. PP is a list of coloured polynomials. The polynomials pairs list is constructed containing those polynomials whose headterm relative to the colouring is defined. The list pairs is in a nondecreasing order. }

procedure PRSCOP (PAIRS: LIST): LIST;
{Pairs copy. PAIRS is a polynomials pairs list. PPAIRS is a copy of PAIRS. }

procedure MKNEWP (P,POL,PRS: LIST): LIST;
{Make new pairs. P is a list of coloured polynomials. POL is a coloured polynomial. PRS is the polynomials pairslist of P. The new pairs between POL and P are constructed and added to PRS. The result is PPAIRS. }

procedure MKCGB (PL: LIST; VAR X,I: LIST);
{Make Comprehensive-Groebner-Basis. PL is a Groebner-System. X is the Comprehensive-Groebner-Basis from PL. I is the number of conditions in PL. }

procedure ADDCGB (PLIST,P: LIST): LIST;
{Add polynomials to comprehensive-groebner-basis. PLIST is list of coloured polynomials. P is a list of polynomials. Those polynomials that are not in P, are added to P without their colouring. the result is PP. }

procedure GSRED (GS: LIST): LIST;
{Groebner-System reduction. GS is a groebner-system. Returns the reduced groebner-system. }

procedure REDUCT (PELEM: LIST): LIST;
{Reduct. PELEM is a pair containing a condition and a polynomials list determined and coloured wrt the condition. The polynomials list is to be reduced. The result together with the condition is R. }

procedure REXTP (P: LIST): LIST;
{Remove extraneous polynomials. P is a list of coloured polynomials. extraneous polynomials relative to their colouring are to be removed. PP is the resulting list. }

procedure RDNORM (COND,FCO,P: LIST; VAR NCO: LIST);
{Reduction normalform. COND is a condition. FCO is a coloured polynomial. P is a list of polynomials determined and coloured by COND. NCO is the normalform of fco modulo P relative to COND. }

procedure REFIND (PCO,P: LIST; VAR PLI,RE: LIST);
{Reduction find polynomial. PCO is a coloured polynomial. P is a list of determined and coloured polynomials. A polynomial for the reduction of PCO modulo P is searched. PLI is emtpy if no polynomial is found. Else PLI is the found polynomial and RE is the term of PCO to be eliminated. }

procedure RMGRT (COND,PP: LIST): LIST;
{Remove green terms. COND is a condition. PP is a list of polynomials determined and coloured relative to COND. If COND contains coefficients to be zero, all green monomials of the polynomials in PP are removed. P is the resulting polynomials list. }

procedure GLOBRE (COND,P: LIST): LIST;
{Global reduction. COND is a condition. P is a list of polynomials. CGB is the coloured polynomials list after global reduction. }

procedure GLEXTP (P: LIST): LIST;
{Global extraneous polynomials remove. P is a list of coloured polynomials. Determined polynomials that are extraneous, are removed. The resulting polynomials list is PP. }

12.8 DIP Arbitrary Domain

procedure DIPEXP (A,NL: LIST): LIST;
{Distributive polynomial exponentiation. D is a non zero distributive polynomial. n is a non-negative beta-integer. B=A**n. If n=0 then a polynomial in zero variables is returned. }

procedure DIFIP (A,D: LIST): LIST;
{Distributive polynomial from distributive integral polynomial. A is a distributive integral polynomial with inverse lexicographical term ordering. D is the domain descriptor for the distributive polynomial B. }

procedure DILRD (V,D: LIST): LIST;
{Distributive polynomial list read. V is a variable list. A list of distributive polynomials in r variables, where r=length(V), r ge 0, is read from the input stream. Any blanks preceding a are skipped. }

procedure DILSUM (A: LIST): LIST;
{Distributive polynomial list sum. D is a circular list of distributive polynomials. B is the sum of all polynomials in A. }

procedure DILWR (A,V: LIST);
{Distributive polynomial list write. V is a variable list. A list of distributive polynomials in r variables, where r=length(V), r ge 0, is written to the output stream. }

procedure DIPBCP (A,BL: LIST): LIST;
{Distributive polynomial base coefficient product. A is a distributive polynomial, b is a base coefficient. C=A*b.}

procedure DIPDIF (A,B: LIST): LIST;
{Distributive polynomial difference. A and B are distributive polynomials. C=A-B.}

procedure DIPFAC (A,VOO: LIST): LIST;
{distributive polynomial factorization. A is a polynomial in distributive representation, VOO is a flag, use variable order optimization iff VOO = 1, returns a list ((e1,f1),...,(ek,fk)), ei positive integers, fi irreducible polynomials in distributive representation, where A = u * f1**e1 * ... * fk**ek and u unit. The ordering of the factors is non-deterministic !! }

procedure DIPIRL (VAR P: LIST; VAR CS: BOOLEAN);
{distributive polynomials interreduced list of polynomials. P is a list of polynomials in distributive representation over an arbitrary domain, CS is a flag, CS = TRUE iff P is changed, returns a interreduced list of polynomials R=(p1,...,pk), R is the result of reducing each pi modulo R-(pi) until no further reductions are possible. }

procedure DIPLIR (P: LIST): LIST;
{distributive polynomial list interreduce. P is a list of polynomials in distributive representation

over an arbitrary domain, returns a interreduced list of polynomials R=(p1,...,pk), R is the result of reducing each pi modulo R-(pi) until no further reductions are possible. }

procedure DIPRLF (P,p: LIST): LIST;
{distributive polynomials reduce list of polynomials with factor. P is a list of polynomials in distributive representation over an arbitrary domain, p is a polynomial of same kind, returns a list of reduced polynomials R=(p1,...,pk), R is the result of reducing each polynomial of P modulo (p) }

procedure DIPMOC (A: LIST): LIST;
{Distributive polynomial monic. A and A are distributive polynomials, C=A/lbc(A) if A ne 0 C=0 if A eq 0. }

procedure DIPNEG (A: LIST): LIST;
{Distributive polynomial negative. B= -A.}

procedure DIPNF (A,B: LIST): LIST;
{distributive polynomial normalform. A is a list of polynomials in distributive representation, B is a polynomial as above, returns a polynomial h such that B is reducible to h modulo A and h is in normalform with respect to A }

procedure DIPQR (A,B: LIST; VAR Q,R: LIST);
{Distributive polynomial quotient and remainder. A and B are distributive polynomials with B ne 0. Q and R are unique distributive rational polynomials such that either B divides A, so Q=A/B and R=0 or B does not divide A, so A=B*Q+R with deg(R) lt deg(B). }

procedure DIPROD (A,B: LIST): LIST;
{Distributive polynomial product. A and B are distributive polynomials. C=A*B.}

procedure DIPS (A,B: LIST): LIST;
{distributive polynomial S-polynomial. A and B are polynomials in distributive representation, returns the S-polynomial of A and B }

procedure DIPSFF (A,VOO: LIST): LIST;
{distributive polynomial squarefree factorization. A is a polynomial in distributive representation, VOO is a flag, use variable order optimization iff VOO = 1, returns a list ((e1,p1),...,(ek,pk)), ei positive integers, pi squarefree polynomials in distributive representation, where A = u * p1**e1 * ... * pk**ek and u unit. }

procedure DIPSUM (A,B: LIST): LIST;
{Distributive polynomial sum. A and B are distributive polynomials. C=A+B. }

procedure DIREAD (V,D: LIST): LIST;
{Distributive polynomial read. V is a variable list. a distributive polynomial A in r variables, where r=length(V), r ge 0, is read from the input stream. any blanks preceding A are skipped. }

procedure DIWRIT (A,V: LIST);
{Distributive polynomial write. A is a distributive polynomial in r variables, r ge 0. V is a variable list for A. A is written in the output stream. }

12.9 DIP Arbitrary Domain Groebner Basis

procedure ADDNFEDIP (f: LIST): LIST;
{Arbitrary domain domain number from extended distributive polynomial. f is an extended distributive polynomial over an arbitrary domain polynomial ring. The domain number of the arbitrary domain is returned. If the unextended distributive polynomial appropriate to f is the zero polynomial then 0 is returned. }

procedure CPEXTEND (f,g: LIST): LIST;
{Critical pair extend. f and g are distributive polynomials such that (f,g) is a critical pair. CPEX-

TEND(f,g) is the appropriate extended critical pair, i.e. the list of f, g and further components which depend on the chosen extended critical pair selection strategy. }

procedure DIPAGB (F: LIST): LIST;
{Distributive polynomial arbitrary domain Groebner basis. F is a list of polynomials in distributive representation. G=DIPAGB(F) is a (not reduced) Groebner basis with Id(G)=Id(F). }

procedure DIPEXTEND (f: LIST): LIST;
{Distributive polynomial extension. f is a polynomial in distributive representation. DIPEXTEND(f) is the extended distributive polynomial appropriate to f. }

procedure DIPRWTDG (f,W: LIST): LIST;
{Distributive polynomial rational-weighted total degree. f is a polynomial in distributive representation in r variables. W=(W1,...,Wr) is a list of rational numbers, where Wi, 1 le i le r, is the weight of the (r-i+1)-th variable in the variable list VALIS, i.e. of the variable w.r.t. the i-th component in each exponent vector of f. DIPRWTDG(f,W) is the rational-weighted total degree of f, defined by the maximal rational-weighted total degree of the exponent vectors of the terms of f. }

procedure ECPINSERT (CP,P: LIST): LIST;
{Extended critical pair insertion. The extended critical pair CP is to be inserted into the extended critical pair list P. ECPINSERT(CP,P) is the list P with the additional pair CP. }

procedure ECPLCMHT (CP: LIST): LIST;
{Extended critical pair select the least common multiple of head terms. ECPLCMHT(CP) is the extended least common multiple of the leadings terms of the two extended polynomials in the extended critical pair CP. }

procedure ECPPOLY1 (CP: LIST): LIST;
{Extended critical pair select the first extended distributive polynomial. }

procedure ECPPOLY2 (CP: LIST): LIST;
{Extended critical pair select the second extended distributive polynomial. }

procedure ECPSELECT (P: LIST; VAR CP,Q: LIST);
{Select an extended critical pair from the extended critical pair list. The extended critical pair list is modified. }

procedure ECPSUGAR (CP: LIST): LIST;
{Extended critical pair select sugar. ECPSUGAR(CP) is the sugar of the S-polynomial of the polynomials in the extended critical pair CP. }

procedure ECPUNEXTEND (CP: LIST): LIST;
{Extended critical pair un-extend. CP is an extended critical pair. ECPUNEXTEND(CP) is the appropriate critical pair, i.e. CP without the extensions which depend on the chosen extended critical pair selection strategy. }

procedure ECPWRITE (CP: LIST);
{Extended critical pair write. The extended critical pair CP is written in the output stream. }

procedure EDIIFSUGNF (P,f: LIST): LIST;
{Extended distributive integral function polynomial normal with sugar strategy normal form. P is a list of non-zero extended distributive integral function polynomials. f is an extended distributive integral function polynomial. EDIIFSUGNF(P,f) is an extended distributive integral function polynomial such that f is reducible to EDIIFSUGNF(P,f) modulo P and EDIIFSUGNF(P,f) is in normalform w.r.t. P. }

procedure EDIIFSUGSP (f,g: LIST): LIST;
{Extended distributive integral function polynomial normal with sugar strategy S-polynomial. f and g are extended distributive integral function polynomials. EDIIFSUGSP(f,g) is the extended S-polynomial of f and g w.r.t. the normal with sugar strategy. }

procedure EDIPEVL (f: LIST): LIST;
{Extended distributive polynomial exponent vector of the leading monomial. f is an extended distributive polynomial. EDIPEVL(f) is the exponent vector of the leading monomial of f. }

procedure EDIPNOR (P,f: LIST): LIST;
{Extended distributive polynomial normal form. P is a list of non-zero extended distributive polynomials. f is an extended distributive polynomial. EDIPNOR(P,f) is an extended distributive polynomial such that f is reducible to EDIPNOR(P,f) modulo P and EDIPNOR(P,f) is in normalform w.r.t. P. }

procedure EDIPSP (f,g: LIST): LIST;
{Extended distributive polynomial S-polynomial. f and g are extended distributive polynomials. EDIPSP(f,g) is the extended S-polynomial of f and g. }

procedure EDIPSUGAR (f: LIST): LIST;
{Extended distributive polynomial sugar. EDIPSUGAR(f) is the sugar of the extended distributive polynomial f. }

procedure EDIPSUGCON (f,S: LIST): LIST;
{Extended distributive polynomial normal with sugar strategy constructor. f is a distributive polynomial. S is the sugar of f. EDIPSUGCON(f,S) is the extended distributive polynomial appropriate to f with sugar S. }

procedure EDIPSUGNOR (P,f: LIST): LIST;
{Extended distributive polynomial normal with sugar strategy normal form. P is a list of non-zero extended distributive polynomials. f is an extended distributive polynomial. EDIPSUGNOR(P,f) is an extended distributive polynomial such that f is reducible to EDIPSUGNOR(P,f) modulo P and EDIPSUGNOR(P,f) is in normalform w.r.t. P. }

procedure EDIPSUGSP (f,g: LIST): LIST;
{Extended distributive polynomial normal with sugar strategy S-polynomial. f and g are extended distributive polynomials. EDIPSUGSP(f,g) is the extended S-polynomial of f and g w.r.t. the normal with sugar strategy. }

procedure EDIPUNEXTEND (f: LIST): LIST;
{Extended distributive polynomial un-extend. f is an extended distributive polynomial. EDIPUNEXTEND(f) is the distributive polynomial appropriate to f. }

procedure EDIPWRITE (f: LIST);
{Extended distributive polynomial write. The extended distributive polynomial f is written in the output stream. }

procedure EVRWTDEG (U,W: LIST): LIST;
{Exponent vector rational-weighted total degree. U=(U1,...,Ur) is an exponent vector. W=(W1,...,Wr) is a list of rational numbers, where Wi, 1 le i le r, is the weight of the (r-i+1)-th variable in the variable list VALIS, i.e. of the variable w.r.t. the i-th component of U. TD=EVRWTDEG(U,W) is the rational-weighted total degree of U, defined by TD = W1U1+...+WrUr. }

procedure INITUPDATE (f: LIST; VAR G,B: LIST);
{The initialization function as a first call of UPDATE. }

procedure LDIPEXTEND (P: LIST): LIST;
{List of distributive polynomials extend. P is a list of distributive polynomials. LDIPEXTEND(P) is the list of the appropriate extended distributive polynomials. }

procedure LECPUNEXTEND (P: LIST): LIST;
{List of extended critical pairs un-extend. P is a list of extended critical pairs. LECPUNEXTEND(P) is the list of the appropriate critical pairs. }

procedure LECPWRITE (P: LIST);
{List of extended critical pairs write. The list P of extended critical pairs is written in the output

stream. }

procedure LEDIPUNEXTEND (P: LIST): LIST;
{List of extended distributive polynomials un-extend. P is a list of extended distributive polynomials. LEDIPUNEXTEND(P) is the list of the appropriate distributive polynomials. }

procedure LEDIPWRITE (P: LIST);
{List of extended distributive polynomials write. The list P of extended distributive polynomials is written in the output stream. }

procedure LRNWRIT (LRN: LIST);
{List of rational numbers write. The list LRN of rational numbers is written in the output stream. }

procedure UPDATE (h: LIST; VAR G,B: LIST);
{Update of extended ideal basis and extended critical pair list as required by DIPAGB. h is an extended distributive polynomial, G is a list of extended distributive polynomials. B is a list of extended critical pairs. }

procedure UpdateVariableWeight ;
{Update of the DIPAGB variable weight list. }

procedure SetDIPAGBOptions (OPT: LIST);
{Set the trace flag, the strategy and the variable weight list of the DIPAGB option record. }

procedure SetDIPAGBStrategy (st: LIST);
{Set the DIPAGB strategy option for the extended critical pair selection. st=0 (= normal) and st=1 (= normal with sugar) are supported. }

procedure SetDIPAGBTraceFlag (tf: LIST);
{Set the DIPAGB trace flag. tf is a non-negative integer level for interactive documentations. }

procedure SetDIPAGBVariableWeight (VW: LIST);
{Set the DIPAGB variable weight list for the normal with sugar strategy. VW is a list of r non-negative rational components where r is the number of variables in the distributive polynomials handled with. The i-th component is the weight of the i-th variable in the variable list VALIS. }

procedure SetCPExtend (EXT: PROCF2);
{Set the critical pair extension function. }

procedure SetDIPExtend (EXT: PROCF1);
{Set the distributive polynomial extension function. }

procedure SetECPInsert (INS: PROCF2);
{Set the extended critical pair insertion function. }

procedure SetECPSelect (SEL: PROCP1V2);
{Set the extended critical pair selection procedure. }

procedure SetECPWrite (WR: PROCP1);
{Set the extended critical pair write procedure. }

procedure SetEDIPNormalform (NF: PROCF2);
{Set the extended distributive polynomial normalform function. }

procedure SetEDIPSPolynomial (SP: PROCF2);
{Set the extended distributive S-polynomial function. }

procedure SetEDIPUnExtend (UEXT: PROCF1);
{Set the extended distributive polynomial un-extension function. }

procedure SetEDIPWrite (WR: PROCP1);
{Set the extended distributive polynomial write procedure. }

procedure SetInit (INIT: PROCP1V2);
{Set the initialization procedure. }

procedure SetUpdate (UPD: PROCP1V2);
{Set the update procedure. }

procedure SetUpdateVariableWeight (UPD: PROCP0);
{Set the DIPAGB variable weight update procedure. }

procedure WriteDIPAGBOptions ;
{Write the current trace flag, strategy and variable weight list of the DIPAGB option record in the output stream. }

procedure WriteDIPAGBStrategy ;
{Write the DIPAGB strategy option for the extended critical pair selection in the output stream. }

procedure WriteDIPAGBTraceFlag ;
{Write the DIPAGB trace flag in the output stream. }

procedure WriteDIPAGBVariableWeight ;
{Write the DIPAGB variable weight list in the output stream. }

12.10 DIP Decompositional Groebner Bases

procedure SetTraceLevel (TL: INTEGER);
{Set Trace-Level for decompositional groebner bases: 0 = no output, except with VOOB, ¿0 = output of time and result after computation, ¿1 = output of messages about tree of computation: number of canceled branches/factors, "cancel factor", "cancel branch", "groebner base", "branch w.o. zeros", ¿2 = output of s-polynomials and normalforms, ¿3 = output of parameters of local procedures and of time and groebner bases during computation. }

procedure SetDecompProc (DCP: INTEGER);
{Set Decomposition-Procedure for decompositional groebner bases: 1 = DIPFAC, 2 = DIPSFF. }

procedure SetUpdateProc (UP: INTEGER);
{Set Update-Procedure for decompositional groebner bases: 1 = UPDATE from module DIPAGB. }

procedure SetVarOrdOpt (VOO: INTEGER);
{Set Variable-Order-Optimization for decompositional groebner bases: 0 = don't optimize, 1 = optimize at begin only, 2 = optimize factorization only, 3 = optimize at begin and factorization }

procedure SetFacSugar (FS: INTEGER);
{Set Factor-Sugar for procedure GroebnerBases1: 0 = sugar of factor is total degree of factor, 1 = sugar of factor is old sugar }

procedure SetReduceExp (RE: INTEGER);
{Set Reduce-Exponent for procedure GroebnerBases2: 1 = reduce (no power of) polynomial ¿1 = reduce corresponding power of polynomial }

procedure SetBranchProc (BP: INTEGER);
{Set Branch-Procedure for procedure GroebnerBases2: 1 = SSCO - new branch for each subset of factors, 2 = EQIEQ - new branch for each factor }

procedure SetDCGBopt (options: LIST);
{Set options for decompositional groebner bases. options is a list of 7 or less elements in following order: 1. Trace-Level (0-4), 2. No. of Decomposition-Procedure (1,2), 3. No. of Update-Procedure (1), 4. Optimization of variable order (0,1,2,3), 5. Sugar of factors for Procedure GroebnerBases1 (0,1), 6. Reduce-Exponent for procedure GroebnerBases2 (¿0), 7. No. of Branch-Procedure in GroebnerBases2 (1,2). }

procedure WriteDCGBopt ;
{write decompositional groebner bases options }

procedure GroebnerBases1 (G: LIST): LIST;
{Distributive polynomials decompositional groebner bases 1. G is a list of polynomials in distributive representation over an arbitrary domain, returns a list (GB1,...,GBk) of groebner bases, where Z(G) = Z(GB1) v ... v Z(GBk). }

procedure GroebnerBases2 (G,U: LIST): LIST;
{Distributive polynomials decompositional groebner bases 2. G and U are lists of polynomials in distributive representation over an arbitrary domain, returns a list ((GB1,U1),...,(GBk,Uk)) of pairs (Gi,Ui), where Gi is a groebner bases, Ui is a list of polynomials and Z(G) n D(U) = (Z(GB1) n D(U1)) v ... v (Z(GBk) n D(Uk)). }

12.11 DIP Domain D-Groebner Bases

procedure DIDPELIMDGB (P : LIST) : LIST;
{Distributive domain polynomial eliminate D-groebner base. P is a list of non zero polynomials in distributive integral representation in r variables. ELIMDGB eliminates the polynominials with respect to the divisibility of the highest monominials. }

procedure DIDPTDR (P, lcmHT, pair : LIST): LIST;
{Distributive domain polynomial top-D-reduzibel. P is a list of non zero polynomials in distributive integral representation in r variables. pair is a pair two integral polynomials in distributive representation. lcmHT is the lcm of the highest terms of the two polynomials. TDR is a boolean value which equals 1, if g is top-D- reduzibel modulo P and 0 if not. }

procedure DIDPCPLMS1 (P : LIST) : LIST;
{Distributive domain polynomial list construct pairs list merge sort. P is a list of non zero polynomials in distributive integral representation in r variables. CPLMS1 sorts a constructed pairs list in the following ascending order: 1. lcm of the highest terms 2. lcm of the highest coefficients P will be changed. }

procedure DIDPLM1 (onestep, twostep : LIST) : LIST;
{Distributive domain polynomial list merge sort. P is a list of non zero polynomials in distributive integral representation in r variables. LM1 merges two (onestep, twostep) constructed pairs lists in the same manner as DIDPLCPLMS1. The lists onestep and twostep will be changed. }

procedure DIDPUCPL1 (P, g, Old : LIST) : LIST;
{Distributive domain polynomial update constructed pairs list. P is a list of integral polynomials in distributive representation. g ist a polynomial in distributive representation. Both are polynomials in r variables. Old is the constructed and sorted pairs list to be updated. }

procedure DIDPGPOL (g1,g2: LIST): LIST;
{Distributive domain polynomial g polynomial. g1 and g2 are integral polynomials in distributive representation. GPOL is the G-polynomial of g1 and g2. }

procedure DIDPSPOL2 (g1, g2, lcmHT, lcmHK: LIST): LIST;
{Distributive domain polynomial s polynomial. g1 and g2 are integral polynomials in distributive representation. lcmHT is the lcm of the highest terms of g1 and g2. lcmHK is the lcm of the highest coefficients of g1 and g2. polynomials in pair. SPol is the S-polynomial of g1 and g2. }

procedure DIDPLEXTAL (AL, g : LIST) : LIST;
{Distributive domain polynomial list extend array list. AL is an array list. g is a polynomial in distributive representation in r variables. Ag is the extended array list of AL. The list AL is modified. }

procedure DIDPLCPL4 (P : LIST; VAR CPL, AL : LIST);
{Distributive domain polynomial list construct pair list. P is a list of polynomials in distributive

representation in r variables. CPL is the constructed pairs list, AL is the Array list. }

procedure DIDPALCMPC (AL, g1, g2, flag : LIST) : LIST;
{Distributive domain polynomial array list check and mark polynomials. AL is an array list. g1, g2 are polynomials in distributive representation in r variables. flag determines whether the pair will be marked as computed or only checked. 1 means to mark 0 only to check. The value 1 is returned if the pair (g1,g2) is already computed otherwise 0 is returned. }

procedure DIDPENF (P,varl,g: LIST): LIST;
{Distributive domain polynomial E-normal form. P is a list of non zero polynomials in distributive integral representation in r variables. g is a distributive integral polynomial. ENF is a polynomial such that g is e-reducible to ENF modulo P and ENF is in E-normalform modulo P. }

procedure DIDPREDDGB (P : LIST) : LIST;
{Distributive domain polynomial reduce D-groebner base. P is a list of non zero polynomials in distributive integral representation in r variables. REDDGB reduces the polynominials. It is nescessary that the highest monomials are pairwise disjoint. }

procedure DIDPSPOL (g1,g2: LIST): LIST;
{Distributive domain polynomial s polynomial. g1 and g2 are integral polynomials in distributive representation. SPol is the S-polynomial of g1 and g2. }

procedure DIDPDNF (P,varl,g: LIST): LIST;
{Distributive domain polynomial D-normal form. P is a list of non zero polynomials in distributive integral representation in r variables. g is a distributive integral polynomial. NF is a polynomial such that g is reducible to NF modulo P and NF is in D-normalform modulo P. }

procedure DIDPDGB (F, TF : LIST): LIST;
{Distributive domain polynomial D-groebner basis. F is a list of integral polynomials in distributive representation in r variables. G is the groebner basis of F. TF the trace flag. }

procedure DIDPEGB (F, DP, TF : LIST): LIST;
{Distributive domain polynomial E-groebner basis. F is a list of integral polynomials in distributive representation in r variables. DP is a domain element with descriptor. G is the groebner basis of F. TF the trace flag. }

12.12 DIP Groebner Bases

procedure DIGMIN (P: LIST): LIST;
{Distributive minimal ordered groebner basis. P is a list of non zero polynomials in distributive representation in r variables. PP is the minimal normed and ordered groebner basis. }

procedure DILIS (P: LIST): LIST;
{Distributive polynomial list irreducible set. P is a list of distributive polynomials, PP is the result of reducing each p element of P modulo P-(p) until no further reductions are possible. }

procedure DIPGB (P,TF: LIST): LIST;
{Distributive polynomial groebner basis. P is a list of polynomials in distributive representation in r variables. PP is the groebner basis of P. tf is the trace flag.}

procedure DIPNOR (P,S: LIST): LIST;
{Distributive polynomial normal form. P is a list of non zero polynomials in distributive representation in r variables. S is a distributive polynomial. R is a polynomial such that S is reducible to R modulo P and R is in normalform with respect to P. }

procedure DIPSP (A,B: LIST): LIST;
{Distributive polynomial S-polynomial. A and B are polynomials in distributive representation. C is the S-polynomial of A and B. }

procedure DIIFGB (P,TF: LIST): LIST;
{Distributive integral function polynomial groebner basis. P is a list of integral function polynomials in distributive representation in r variables. PP is the groebner basis of P. tf is the trace flag.}

procedure DIIFLS (P: LIST): LIST;
{Distributive integral function polynomial list irreducible set. P is a list of distributive integral function polynomials, PP is the result of reducing each p element of P modulo P-(p) until no further reductions are possible. }

procedure DIIFMI (P: LIST): LIST;
{Distributive minimal ordered groebner basis. P is a list of non zero integral function polynomials in distributive representation in r variables. PP is the minimal normed and ordered groebner basis. }

procedure DIIFNF (P,RPP,S: LIST): LIST;
{Distributive integral function polynomial normal form. P is a list of non zero polynomials in distributive integral function representation in r variables. S is a distributive integral function polynomial. R is a polynomial such that S is reducible to R modulo P and R is in normalform with respect to P. }

procedure DIIFSP (A,B: LIST): LIST;
{Distributive integral function polynom S-polynomial. A and B are integral function polynomials in distributive representation. C is the S polynomial of A and B. }

12.13 Distributive Polynomials Tools

procedure VVECFVLIST (v1,v2:LIST):LIST;
{variable vector from variable lists. v1,v2 are variable lists, such that v1 is contained in v2. A variable vector representing v1 w.r.t. v2 is returned. }

procedure VVECC (v:LIST):LIST;
{variable vector complement. v is a variable vector. The complement of v is returned. }

procedure DIPONE (d:LIST):LIST;
{distributive polynomial arbitrary domain one. The polynomial 1 in the actual polynomial ring is returned. The polynomial ring is determined by the global variable VALIS and the domain element d. }

procedure ADDDFDIP (p:LIST):LIST;
{arbitrary domain domain descriptor from distributive polynomial. p is a polynomial over an arbitrary domain polynomial ring. The domain descriptor of the arbitrary domain is returned. If p=0, then 0 is returned. }

procedure ADDDFDIPD (p,d:LIST):LIST;
{arbitrary domain domain descriptor from distributive polynomial or default. p is a polynomial over an arbitrary domain polynomial ring, d is a domain descriptor. The domain descriptor of the arbitrary domain is returned. If p is the zero polynomial, then the default value d is returned. }

procedure ADDDFDIL (l:LIST):LIST;
{arbitrary domain domain descriptor from distributive polynomial list. l is a list of polynomials over an arbitrary domain polynomial ring. The domain descriptor of the arbitrary domain is returned. If l is the empty list or all polynomials in l are equal to zero, then 0 is returned. }

procedure ADDDFDILD (l,d:LIST):LIST;
{arbitrary domain domain descriptor from distributive polynomial list or default. l is a list of polynomials over an arbitrary domain polynomial ring, d is a domain descriptor. The domain descriptor of the arbitrary domain is returned. If l is the empty list or all polynomials in l are equal to zero, then 0 is returned. }

procedure ADDNFDIP (p:LIST):LIST;
{arbitrary domain domain number from distributive polynomial. p is a polynomial over an arbitrary domain polynomial ring. The domain number of the arbitrary domain is returned. If p is the zero polynomial, then 0 is returned. }

procedure ADDNFDIPD (p,d:LIST):LIST;
{arbitrary domain domain number from distributive polynomial or default. p is a polynomial over an arbitrary domain polynomial ring, d is a domain number. The domain number of the arbitrary domain is returned. If p is the zero polynomial, then the default value d is returned. }

procedure ADDNFDIL (l:LIST):LIST;
{arbitrary domain domain number from distributive polynomial list. l is a list of polynomials over an arbitrary domain polynomial ring. The domain number of the arbitrary domain is returned. If l is the empty list or all polynomials in l are equal to zero, then 0 is returned. }

procedure ADDNFDILD (l,d:LIST):LIST;
{arbitrary domain domain number from distributive polynomial list or default. l is a list of polynomials over an arbitrary domain polynomial ring, d is a domain number. The domain number of the arbitrary domain is returned. If l is the empty list or all polynomials in l are equal to zero, then 0 is returned. }

procedure DIPPOWER (p,n:LIST):LIST;
{distributive polynomial power. p is a distributive polynomial over an arbitrary domain, n is an integer. The polynomial p**n is returned. }

procedure DILPROD (L:LIST;domain:LIST):LIST;
{distributive polynomial list product. L is a list of distributive polynomials over an arbitrary domain. The product of all polynomials in L is returned. The variable domain specifies the domain of the polynomial ring. This is necessary to form the polynomial 1, if the list L is empty. }

procedure DIPDEGI (p,i:LIST):LIST;
{distributive polynomial degree of i-th main variable. p is a distributive polynomial in r variables. 0¡i¡=r is an atom. The degree of the i-th variable is returned. The variable are numbered accordingly to their occurrence in VALIS. }

procedure DILMOC (L:LIST):LIST;
{distributive polynomial monic. L is a list of distributive polynomials over an arbitrary domain. Each polynomial is normalized in such a way, that its highest coefficient is 1. Note, the inverses of the highest coefficients must exist. Identical polynomials are deleted. }

procedure DIPPAD (p,i: LIST):LIST;
{distributive polynomial partial derivation. p is a distributive polynomial in r variables. 0¡i¡=r is an atom. $\frac{\partial p}{\partial X\_i}$ is returned. X_i is the i-th element in the variable list VALIS }

procedure DIMPAD (c,ev,i:LIST):LIST;
{distributive monomial partial derivation. c is a arbitrary domain element. e is an exponent vector with r elements. 0¡i¡=r is an atom. $\frac{\partial cX\hat{}ev}{\partial X\_i}$ is returned. X_i is the i-th element in the variable list VALIS The result is returned as an distributive polynomial. }

procedure DIPCPP (P:LIST; VAR content,ppt: LIST);
{distributive polynomial content and primitive part. P is a distributive polynomial over an arbitrary domain. The following domain functions must be set: ADSIGN, ADONE, ADNEG, ADQUOT, ADGCD. The content of p is stored in content and the primitive part of P is stored in ppt. }

procedure DIPPCPP (P:LIST; VAR content,ppt: LIST);
{distributive polynomial pseudo content and primitive part. P is a distributive polynomial over an arbitrary domain. The following domain functions must be set: ADONE, ADNEG, ADQUOT. ppt is a monic polynomial. }

procedure DIPCNST (dip:LIST): BOOLEAN;
{distributive polynomial is constant. dip is a distributive polynomial. True is returned iff the polynomial is constant, i.e. there is only one monomial and the exponent vector of the monomial is a tuple containing only zeroes.. }

procedure DIPCNSTR (p,v: LIST):BOOLEAN;
{distributive polynomial constant relative to variables. p is a distributive polynomial in n variables. v is a variable vector. True is returned, iff p is constant w.r.t. to the variables in v. }

procedure EVCNSTR (ev,mvars:LIST):BOOLEAN;
{exponent vector constant relatively. ev is a exponent vector of length r. mvars is a variable vector of length r. True is returned, iff ev is constant relatively to mvars. }

procedure EvordPush (evord: LIST);
{evord push. evord is a value for the global variable EVORD of the module DIPC. The variable EVORD is set to evord. The old value of EVORD is stored on the to of the evord stack. It can be restored with the command PopEvord. }

procedure EvordPop ();
{evord pop. The global variable EVORD is set to the top element of the evord stack. The top element is deleted. }

procedure ValisPush (valis: LIST);
{valis push. valis is a value for the global variable VALIS of the module DIPC. The variable VALIS is set to valis. The old value of VALIS is stored on the to of the valis stack. It can be restored with the command PopValis. }

procedure ValisPop ();
{valis pop. The global variable VALIS is set to the top element of the valis stack. The top element is deleted. }

procedure DILINV (dil,j,k:LIST):LIST;
{distributive polynomial list introduce new variable. dil is a list of polynomials in a polynomial ring R(X1,...,Xr), 0¡=j¡r, k¿0. The polynomials are transfered into the polynomial ring R(X1,...,Xj,Y1,...,Yk,Xj+1,...,Xr). Be carefully, the EV of the original polynomials is (Xr,...,X1) and the EV of the new polynomials is (Xr,...,Xj+1,Yk,...,Y1,Xj,...,X1). So j+1,... are the positions of the new variables in VALIS but not in the exponent vectors of the polynomials. }

procedure DIPFDIPP (p,NewDd:LIST; VAR q, vlist: LIST);
{distributive polynomial from distributive polynomial over polynomial ring. p is a distributive polynomial over an arbitrary domain d, which is a (represented recursively) polynomial ring R[U] (or more exactly R[U1][U2]...). A distributive polynomial over the domain with the domain descriptor NewDd is returned in the variable q, the varlist for q in vlist. It is supposed that the coefficients of the polynomial p have the form (id,pp,r,valis) where id is the domain identifier, pp is the polynomial, r is the number of variables and valis is the variable list for pp. No adaption to the domain elements are done. The global variable VALIS must be set. }

procedure EVEXT (p,evx:LIST):LIST;
{exponent vector extension. p is distributive polynomial not equal to zero. exv is an exponent vector. All exponent vectors of monomials of p are extended with evx, i.e. each exponent vector ev is replaced with CONC(evx,ev). }

procedure ADPFDIP (p, dd: LIST): LIST;
{arbitrary domain polynomial from distributive polynomial. p is a distributive polynomial. dd is an arbitrary domain descriptor. All coefficients of p are casted to the domain dd. The result is returned. }

procedure DIPPFDIP (p,r,NewDd:LIST;VAR q,vlist: LIST);
{distributive polynomial over polynomial ring from distributive polynomial. p is distributive polynomial over an arbitrary domain R. 0¡r¡DIPNOV(p) is a number of variables. NewDd is a domain

descriptor. The representation of the polynomial p is changed. Let p in R[U1,...,Ur,X1,...Xn]. The polynomial p is represented as an element in the polynomial ring R[U1,...,Um][X1,...,Xn]. NewDd must be the domain descriptor of the Ring R[U1,...,Um]. Polynomials in the ring R[U1,...,Um] must be represented recursively. The new representation of the polynomial p is returned in q. The new list of main variables is returned in vlist. The global variable VALIS must be set. }

procedure MPPFMP (Coeff,Ev,r:LIST;VAR RCpol,NewEv: LIST);
{monomial of polynomial ring over polynomial ring from monomial of polynomial ring. Coeff is the coefficient of the monomial, Ev describes the term of the monomial. r is the number of variables, which are shifted into the coefficient ring. RCpol is set to the recursive represented coefficient polynomial. NewEv is the term of the new monomial. }

procedure DIPCONV (p,E: LIST):LIST;
{distributive polynomial conversion. p is a distributive polynomial over an arbitrary domain D. All coefficients of p are converted to the domain E. It is necessary, that the conversion function from the domain D to the domain E is available. (Set this function with SetConvFunc(D,E,f1)) }

procedure DILCONV (P,E: LIST):LIST;
{distributive polynomial list conversion. P is a list of distributive polynomials p. Each p is a distributive polynomial over an arbitrary domain D. All coefficients of p are converted to the domain E. It is necessary, that the conversion function from the domain D to the domain E is available. (Set this function with SetConvFunc(D,E,f1)) }

procedure DIPFADIP (p: LIST):LIST;
{distributive polynomial from arbitrary domain integral polynomial. p is an element of the arbitrary domain IP (u_1,...,u_r). The polynomial p is returned represented as an distributive polynomial over the arbitrary domain INT.}

procedure DIPFIP (p,r: LIST):LIST;
{distributive polynomial from integral polynomial. p is an integral polynomial, r is the number of variables of p. The polynomial p is returned represented as an distributive polynomial over the arbitrary domain INT.}

procedure DILPFDIL (L,r,newdd:LIST):LIST;
{distributive polynomials over polynomial ring list from distributive polynomial list. L is a list of distributive polynomials, r is the number of variables that are shifted in the coefficient ring, newdd is the domain descriptor for the new coefficient. For each polynomial in l the first r variables of the polynomial ring are shifted in the coefficient ring. }

procedure DILFDILP (L,NewDd:LIST):LIST;
{distributive polynomial list from distributive polynomial list over polynomial ring list. L is a list of distributive polynomials over an polynomial ring. NewDd is a domain descriptor. All variables of the coefficient ring are shifted to the main variables. NewDd is the domain descriptor for the new coefficient ring. }

procedure DIPCT (p: LIST): LIST;
{distributive polynomial coefficient tuple. p is a univariate distributive polynomial over an arbitrary domain. The coefficient tuple (a_0,...,a_d) of p is returned. }

procedure DIPIMO (p:LIST):LIST;
{distributive polynomial inverse monomial order. p is a distributive polynomial. The order of the monomials in the polynomials p is inverted. The polynomial p is modified. The result is returned. }

procedure DILIMO (P:LIST):LIST;
{distributive polynomial list inverse monomial order. P is a list of distributive polynomials p. The order of the monomials in each p is inverted. Each p is modified to obtain the result. }

procedure DIPXCM (p,mvars: LIST):LIST;
{distributive polynomial extract constant monomials. p is a distributive polynomial in r variables.

mvars is a variable vector. A polynomial is returned. This polynomial contains all monomials of p which are constant w.r.t. mvars. }

procedure DIPMVV (p: LIST):LIST;
{distributive polynomial minimal variable vector. p is a distributive polynomial. A variable vector containing all variables occuring in p is returned. }

12.14 MAS Domain Algebraic Number

procedure DomLoadAF ();
{Domain load algebraic number. }

12.15 MAS Domain Arbitrary Precision Floating Point

procedure DomLoadAPF ();
{Domain load arbitrary precision floating point. }

12.16 MAS Domain Complex Number

procedure DomLoadC ();
{Domain load complex number. }

12.17 MAS Domain Finite Field

procedure DomLoadFF ();
{Domain load finite field. }

12.18 MAS Domain Integer

procedure DomLoadI ();
{Domain load integer. }

12.19 MAS Domain Integral Polynomial

procedure DomLoadIP ();
{Domain load integral polynomials . }

12.20 MAS Domain Modular Digit

procedure DomLoadMD ();
{Domain load modular digit. }

12.21 MAS Domain Modular Integer

procedure DomLoadMI ();
{Domain load modular integer. }

12.22 MAS Domain Octonion Number

procedure DomLoadO ();
{Domain load octonion number. }

12.23 MAS Domain Quaternion Number

procedure DomLoadQ ();
{Domain load quaternion number. }

12.24 MAS Domain Rational Function

procedure DomLoadRF ();
{Domain load rational function. }

12.25 MAS Domain Rational Number

procedure DomLoadRN ();
{Domain load rational number. }

12.26 MAS Domain Rational Polynomial

procedure DomLoadRP ();
{Domain load rational polynomials . }

12.27 MAS Arbitrary Domain

procedure ADABSF (A:LIST):LIST;
{Arbitrary domain absolute value. }

procedure ADCNST (A: LIST): BOOLEAN;
{Arbitrary domain constant test. Returns true iff A is a constant. }

procedure ADCOMP (A,B:LIST):LIST;
{Arbitrary domain comparison. c=sign(a-b). }

procedure ADCONV (A,B: LIST): LIST;
{Arbitrary domain conversion. c=b:domain(a). }

procedure ADDIF (A,B: LIST): LIST;
{Arbitrary domain difference. c=a-b. }

procedure ADEXP (A,NL: LIST): LIST;
{Arbitrary domain exponentiation. c=a**nl. }

procedure ADFACT (A: LIST): LIST;
{Arbitrary domain factorization. Returns a list containing all prime factors of A.}

procedure ADFACTO (A: LIST): LIST;
{Arbitrary domain factorization with variable order optimization. A is an arbitrary domain polynomial. Returns a list containing all prime factors of A.}

procedure ADFI (D,A: LIST): LIST;
{Arbitrary domain from integer. D is a domain element and A is an integer. }

procedure ADFIP (D,A: LIST): LIST;
{Arbitrary domain from integral polynomial. D is a domain element and A is an integral polynomial in #vldd(D) variables. }

procedure ADGCD (A,B: LIST): LIST;
{Arbitrary domain greatest common divisor. c=gcd(a,b). }

procedure ADGCDC (A,B: LIST; VAR C,AA,BB: LIST);
{Arbitrary domain greatest common divisor and cofactors. C=gcd(A,B), A=C*AA, B=C*BB, if C=0 then AA=BB=0. If gcd is undefined for the current domain then C:=1, AA:=A, BB:=B. }

procedure ADGCDE (A,B: LIST; VAR C,AA,BB: LIST);
{Arbitrary domain greatest common divisor and linear combination. C=gcd(A,B), C=A*AA+B*BB. If gcd is undefined for the current domain then C:=1, AA:=0, BB:=0. }

procedure ADINV (A: LIST): LIST;
{Arbitrary domain inverse. c=1/a. }

procedure ADINVT (A: LIST): LIST;
{Arbitrary domain inverse existence test. tl=1 if a is invertible, tl=0 else. }

procedure ADLCM (A,B: LIST): LIST;
{Arbitrary domain least common multiple. c=lcm(a,b). }

procedure ADNEG (A: LIST): LIST;
{Arbitrary domain negative. c=-a. }

procedure ADONE (A: LIST): LIST;
{Arbitrary domain one. sl=1 if a=1, sl ne 1 else. }

procedure ADPCPP (P: LIST; VAR c, pp: LIST);
{Arbitrary domain polynomial content and primitive part. P is a distributive polynomial over an arbitrary domain. The content of c and its primitive part is returned. It holds P=c * pp. If the domain is a field then HC(pp)=1. If ADSIGN is defined in the domain, then ADSIGN(HC(p))¿=0. }

procedure ADPROD (A,B: LIST): LIST;
{Arbitrary domain product. c=a*b. }

procedure ADQR (A,B:LIST; VAR Q,R:LIST);
{Arbitrary domain quotient and remainder. q=a/b, r=a-(a/b)*b. }

procedure ADQUOT (A,B: LIST): LIST;
{Arbitrary domain quotient. c=a/b. }

procedure ADREAD (D: LIST): LIST;
{Arbitrary domain read. d is the domain descriptor. }

procedure ADREM (A,B:LIST):LIST;
{Arbitrary domain remainder. r=a-(a/b)*b. }

procedure ADSIGN (A: LIST): LIST;
{Arbitrary domain sign. cl=sign(a). }

procedure ADSUM (A,B: LIST): LIST;
{Arbitrary domain sum. c=a+b. }

procedure ADTOIP (A: LIST; VAR LCM: LIST): LIST;
{Arbitrary domain to integral polynomial conversion. LCM is the lcm of coefficient-denominators }

procedure ADWRIT (A: LIST);
{Arbitrary domain write. }

procedure ADDDREAD (): LIST;
{Arbitrary domain, domain descriptor read. A domain element with descriptor D is read from the input stream. }

procedure ADDDWRIT (D: LIST);
{Arbitrary domain, domain descriptor write. d is a domain element with descriptor. d is written to the output stream. }

procedure ADVLDD (D: LIST): LIST;
{variable list from domain descriptor. d is a domain element with descriptor. if the domain depends on some variables, then the related variable list is returned, otherwise the empty list is returned. }

procedure SetAbsFunc (d: Domain; f1: PROCF1);
{Set absolute value function in domain. d is a domain and f1 is a function of one LIST argument. }

procedure SetCnstFunc (d: Domain; f1: PROCF1B);
{Set constant test function in domain. d is a domain and f1 is a boolean-function of one LIST argument. }

procedure SetCompFunc (d: Domain; f2: PROCF2);
{Set comparison function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetConvFunc (d1, d2: Domain; f1: PROCF1);
{Set conversion function in domain. d1 and d2 are domains and f2 is a function of two LIST arguments. }

procedure SetDifFunc (d: Domain; f2: PROCF2);
{Set difference function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetExpFunc (d: Domain; f2: PROCF2);
{Set exponential function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetFactFunc (d: Domain; f1: PROCF1);
{Set factorization function in domain. d is a domain and f1 is a function of one LIST argument. }

procedure SetFIntFunc (d: Domain; f2: PROCF2);
{Set from integer function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetFIPolFunc (d: Domain; f2: PROCF2);
{Set from integral polynomial function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetGcdFunc (d: Domain; f2: PROCF2);
{Set gcd function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetGcdcFunc (d: Domain; p2v3: PROCP2V3);
{Set gcd-and-cofactors function in domain. d is a domain and p2v3 is a procedure of 2 LIST and 3 VAR LIST arguments.}

procedure SetGcdeFunc (d: Domain; p2v3: PROCP2V3);
{Set gcd-and-lin-combination function in domain. d is a domain and p2v3 is a procedure of 2

LIST and 3 VAR LIST arguments.}

procedure SetInvFunc (d: Domain; f1: PROCF1);
{Set inversion function in domain. d is a domain and f is a function of one LIST argument. }

procedure SetInvTFunc (d: Domain; f1: PROCF1);
{Set inversion test function in domain. d is a domain and f is a function of one LIST argument. }

procedure SetLcmFunc (d: Domain; f2: PROCF2);
{Set lcm function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetNegFunc (d: Domain; f1: PROCF1);
{Set negation function in domain. d is a domain and f is a function of one LIST argument. }

procedure SetOneFunc (d: Domain; f1: PROCF1);
{Set one test function in domain. d is a domain and f is a function of one LIST argument. }

procedure SetPCppFunc (d:Domain; p1v2: PROCP1V2);
{Set Content and primitive part function. d is a domain and and p2v3 is a procedure of 2 LIST and 3 VAR LIST arguments.}

procedure SetProdFunc (d: Domain; f2: PROCF2);
{Set product function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetQrFunc (d: Domain; p2v2: PROCP2V2);
{Set quotient and remainder function in domain. d is a domain and p2v2 is a procedure of 2 LIST and 2 var LIST arguments. }

procedure SetQuotFunc (d: Domain; f2: PROCF2);
{Set quotient function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetReadFunc (d: Domain; f1: PROCF1);
{Set read function in domain. d is a domain and f1 is a function of one LIST argument. }

procedure SetRemFunc (d: Domain; f2: PROCF2);
{Set remainder function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetSignFunc (d: Domain; f1: PROCF1);
{Set sign function in domain. d is a domain and f1 is a function of one LIST argument. }

procedure SetSumFunc (d: Domain; f2: PROCF2);
{Set sum function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetToipFunc (d: Domain; f1v1: PROCF1V1);
{Set conversion-to-integer-polynomial function in domain. d is a domain and f1v1 is a function of one LIST and one VAR LIST argument. }

procedure SetWritFunc (d: Domain; p1: PROCP1);
{Set write function in domain. d is a domain and p1 is a procedure of one LIST argument. }

procedure SetVlddFunc (d: Domain; f1: PROCF1);
{Set variable list from domain descriptor function in domain. d is a domain and f is a function of one LIST argument. }

procedure SetDdrdFunc (d: Domain; f0: PROCF0);
{Set domain descriptor read function in domain. d is a domain and f0 is a function with no arguments. }

procedure SetDdwrFunc (d: Domain; p1: PROCP1);
{Set domain descriptor write function in domain. d is a domain and p1 is a procedure of one LIST argument. }

procedure NewDom (S, s: ARRAY OF CHAR): Domain;
{New domain. S is a domain identificator and s is a domain name. A new domain is returned. }

procedure DomSummary ();
{Arbitrary domain summary. A summary of all defined domains is written to the output stream. }

procedure ADPFACT (P,VOO: LIST): LIST;
{Arbitrary domain polynomial factorization. P is a polynomial in distributive representation over an arbitrary domain, VOO is a flag, use variable order optimization iff VOO = 1, returns the list ((e1,f1),...,(ek,fk)), ei positive integers, fi irreducible polynomials in distributive representation, where P = u * f1**e1 * ... * fk**ek and u unit. }

procedure ADPNF (G,P: LIST): LIST;
{Arbitrary domain polynomial normalform. G is a list of polynomials in distributive representation over an arbitrary domain, P is a polynomial as above, returns a polynomial h such that P is reducible to h modulo G and h is in normalform with respect to G }

procedure ADPSFF (A,VOO: LIST): LIST;
{Arbitrary domain polynomial squarefree factorization. A is a polynomial in distributive representation over an arbitrary domain, VOO is a flag, use variable order optimization iff VOO = 1, returns a list ((e1,p1),...,(ek,pk)), ei positive integers, pi squarefree polynomials in distributive representation, where A = u * p1**e1 * ... * pk**ek and u unit. }

procedure ADPSP (A,B: LIST): LIST;
{Arbitrary domain polynomial S-polynomial. A and B are polynomials in distributive representation over an arbitrary domain, returns the S-polynomial of A and B }

procedure ADPSUGNF (G,P: LIST): LIST;
{Arbitrary domain polynomial normal with sugar strategy normalform. G is a list of extended polynomials in distributive representation over an arbitrary domain, P is an extended polynomial as above, returns an extended polynomial h such that P is reducible to h modulo G and h is in normalform with respect to G }

procedure ADPSUGSP (A,B: LIST): LIST;
{Arbitrary domain polynomial normal with sugar strategy S-polynomial. A and B are extended polynomials in distributive representation over an arbitrary domain, returns the extended S-polynomial of A and B }

procedure SetPFactFunc (d: Domain; f1: PROCF1);
{Set factorization function in domain. d is a domain and f1 is a function of one LIST argument. }

procedure SetFactoFunc (d: Domain; f1: PROCF1);
{Set factorization with variable order optimization function in domain. d is a domain and f1 is a function of one LIST argument. }

procedure SetPNormFunc (d: Domain; f2: PROCF2);
{Set polynomial normalform function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetPSqfrFunc (d: Domain; f1: PROCF1);
{Set polynomial squarefree factorization function in domain. d is a domain and f1 is a function of one LIST argument. }

procedure SetPSpolFunc (d: Domain; f2: PROCF2);
{Set polynomial S-polynomial function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetPSugNormFunc (d: Domain; f2: PROCF2);
{Set polynomial normal with sugar strategy normalform function in domain. d is a domain and f2 is a function of two LIST arguments. }

procedure SetPSugSpolFunc (d: Domain; f2: PROCF2);
{Set polynomial normal with sugar strategy S-polynomial function in domain. d is a domain and

f2 is a function of two LIST arguments. }

Chapter 13

Counting Real Roots

13.1 Linear algebra definition module

procedure ADUM (D,n: LIST): LIST;
{Arbitrary domain unit matrix. n is an integer. The (n x n) unit matrix of domain D is returned. }

procedure ADVSPROD (D,A,B: LIST): LIST;
{Arbitrary domain vector scalar product. A and B are vectors of the domain D. The arbitrary domain value C = a1*b1 + ... + an*bn is returned. }

procedure ADVSVPROD (A,b: LIST): LIST;
{Arbitrary domain vector scalar vector product. A is an arbitrary domain vector and b is a number of the same domain. The arbitrary domain vector C = (a1*b, ..., an*b) is returned. }

procedure ADVVSUM (A,B: LIST): LIST;
{Arbitrary domain vector vector sum. A and B are arbitrary domain vectors. The arbitrary domain vector C = (a1+b1, ..., an+bn) is returned. }

procedure ADSMPROD (A,b: LIST): LIST;
{Arbitrary domain scalar and matrix product. A is a arbitrary domain matrix. b is a arbitrary domain number. The arbitrary domain matrix C = A * b is returned. }

procedure ADMSUM (A,B: LIST): LIST;
{Arbitrary domain matrix sum. A and B are arbitrary domain matrices. The arbitrary domain matrix C = A + B is returned. }

procedure ADMPROD (D,A,B: LIST): LIST;
{Arbitrary domain matrix product. A and B are matrices of domain D. The matrix C = A * B of domain D is returned, if the number of columns of A is equal to the number of rows of B, otherwise the empty matrix is returned. }

procedure ADVWRITE (A: LIST);
{Arbitrary domain vector write. A is an arbitrary domain vector. A is written to the output stream. }

procedure ADMWRITE (A: LIST);
{Arbitrary domain matrix write. A is an arbitrary domain matrix. A is written to the output stream. }

procedure ADMTRACE (D,A: LIST): LIST;
{Arbitrary domain matrix trace. A is a matrix of domain D. The trace of A is returned. }

procedure ADMPTRACE (D,A,B: LIST): LIST;
{Arbitrary domain matrix product trace. A and B are matrices of domain D. The trace of A*B is returned. }

procedure ADCHARPOL (D,Q: LIST): LIST;
{Arbitrary domain characteristic polynomial. Q is a p x p Matrix of domain D. The list al=(a(0),...,a(p)) is created such that a(i) from D is the coefficient of X^(p-i) in det(XE-Q). }

procedure ADSIG (D,Q: LIST): LIST;
{Arbitrary domain signature. Q is a symmetric p x p Matrix of domain D. The signature of Q ist returned. ADCHARPOL is used. }

procedure IMTRACE (A: LIST): LIST;
{Integral matrix trace. A is an integral matrix. The trace of A is returned. }

procedure IMPTRACE (A,B: LIST): LIST;
{Integral matrix product trace. A and B are integral matrices. The trace of the matrix A*B is returned. }

procedure ICHARPOL (Q: LIST): LIST;
{Integral matrix characteristic polynomial. Q is an integral p x p Matrix. The list al = (a(0),...,a(p)) of integers is created with a(i) is the coefficient of X^(p-i) in det(XE-Q). }

procedure ISIG (Q: LIST): LIST;
{Integral matrix signature. Q is a symmetric integral p x p Matrix. The signature of Q ist returned. ICHARPOL is used }

procedure IMRTPROD (A,B: LIST): LIST;
{Integral matrix right tensor product. A and B are integral matrices. The matrix C is constructed by replacing every entry a(i,j) of A by the matrix a(i,j)*B. }

13.2 Real Root Arbitrary Domain

procedure EVUMSORT (L: LIST): LIST;
{Exponent vector unique merge sort. The list of exponent vectors L ist sorted with respect to the actual termorder. Multiple entries are deleted. }

procedure EVSSPROD (T: LIST): LIST;
{Exponent vektor set sorted product. T is a list of terms. U = a*b: a,b from T is sorted with respect to the actual termorder. }

procedure RRVTEXT (A,L: LIST): LIST;
{Real root vector of tupels extension. A is a set of s-tupels, L is a set of objects. B is a set of (s+1)-tupels constructed by appending each object of L to the tupels of A. The ordering of B is increasing lexicographical wrt the ordering of L and A. }

procedure RRZDIM (G: LIST): LIST;
{Real root zero-dimensional test. G non-trivial reduced groebner basis. s = 1 iff Id(G) is zero-dimensional, s = 0 otherwise. }

procedure RRREDTEST (G,t: LIST; VAR g,s: LIST);
{Real root reducibility test. G reduced groebner basis, t term. s = 0, if t is reducible with an Element g of G with HT(g) = t, s = -1, if t is not reducible and s = 1 otherwise }

procedure RRREDTERMS (G: LIST): LIST;
{Real root reduced terms. G reduced groebner basis of a nontrival zerodimensional ideal. R ist the set of reduced terms sorted with respect to the actual termorder. }

procedure RRADNFORM (D,G,R: LIST): LIST;
{Real root arbitrary domain normal form. G monic reduced groebner basis of a nontrivial zerodi-

mensional ideal of domain D, R is the set of reduced terms. NF is a list of entries (u,ut,up) with: u is an element of R * R, RRREDTEST(G,u,_,ut) and up is the normal form of u wrt G. The elements of NF are sorted with respect to the actual termorder in decreasing order of the first entry. }

procedure RRADSTRCONST (D,G,R: LIST; VAR U, beta: LIST);
{Real root arbitrary domain structure constants. G monic reduced groebner basis of a nontrivial zerodimensional ideal of domain D, R is the set of reduced terms. beta is a matrix of structure-constants beta[u,v] wrt the basis R with u from U = R * R and v from R. U and the rows of the matrix beta are sorted with respect to the actual termorder in increasing order. }

procedure RRMMULT (w,R,U,beta: LIST): LIST;
{Real root matrix of multiplication. R is the set of reduced terms in increasing order, w from R, U = R * R and beta is the set of combined structure constants wrt R. The columnwise represented matrix of multiplication with w is returned. }

procedure RRADVARMATRICES (G,R,U,beta: LIST): LIST;
{Real root arbitrary domain multiplication matrices of variables. G monic reduced groebner basis of a nontrivial zerodimensional ideal. R is the set of reduced terms in increasing order, U = R * R and beta is the set of combined structure constants wrt R. L is a list of entries of the form (1,M(i)) and M(i) is the matrix of multiplication with X(i) wrt R. }

procedure RRADPOLMATRIX (D,R,h: LIST; VAR Mh,L: LIST);
{Real root arbitrary domain polynomial matrix. R is the set of reduced terms in increasing order, h is a polynomial of domain D, L contains nonempty lists L(i) of the form j(1),M(1),j(2),M(2),...,j(k),M(k) with 1=j(1)¡j(2)¡...¡j(k) and M(l) ist the matrix of multiplikation with X(i)**j(l) for the variable X(i). L is extended with new calculated matrices. Mh is the matrix of multiplication with h. }

procedure RRADQUADFORM (D,R,U,beta,Mh: LIST): LIST;
{Real root arbitrary domain quadratic form. D is a domain, R is the set of reduced terms in increasing order, U = R * R and beta is the set of combined structure constants wrt R. Mh is the matrix of multiplication with h represented columnwise. The matrix Q=(q(i,j)) with q(i,j) = tr(M(h)*M(v(i))*M(v(j)) and v(i),v(j) from R is computed. }

procedure RRCSR (i,v,tf: LIST; VAR A,N,S,L: LIST): LIST;
{Real root count solve and reduce. Subroutine of the RR*COUNT procedures. i is an integer, v is a sign vector, tf is the trace-flag. A is an integral k x k- matrix, N is a subset of -1,0,+1**i of length k sorted in increasing lexicographical order, S is an integral vector of length k and L is a list of length k. The system A*Z=S is solved and reduced by deleting the zero entries of Z and the corresponding columns of A. Then the system is reduced to a regular one by deleting linear dependend rows of A and the corresponding elements of S and L. ZNL is a list of pairs (z,n) with z ¿ 0 is an element of Z and n is the corresponding element of N. If there does not exist an element n in N satisfiing the sign condition v then the empty list is returned. }

procedure RRADCOUNT (D,G,H,v,tf: LIST): LIST;
{Real root arbirary domain count. G is a monic reduced groebner basis of a zerodimensional ideal of domain D, H is a list of polynomials of length s. v is a vektor of signs with length not greater than s. tf is the trace-flag. ZNL is a list of pairs (z,n) with n is an element of -1,0,+1**s and z ¿ 0 is the number of real zeroes of G wrt the sign condition n for the elements of H. ZNL is sorted wrt the invers lexicographical order of the n. If there does not exist any real zero or a zero satisfiing the sign condition v, then the empty list is returned. }

13.3 Real Root Integral

procedure RRIPIQ (c: LIST; VAR A,a: LIST);
{Real root integral polynomial integral quotient. A is an integral polynomial. a and c are nonzero

integers. New values for A and a are computed such that the equation A/a := (1/c)*(A/a) holds. If a and the content of A have gcd 1 then this is also true for the result. }

procedure RRIPQSUM (B,b,c: LIST; VAR A,a: LIST);
{Real root integral polynomial quotient sum. A and B are integral polynomials. a,b and c are non zero integers. New values for A and a are computed such that the equation A/a := A/a + c*(B/b) holds. If b and the content of B have gcd 1 and a and the content of A have gcd 1 then the new integral polynomial A has gcd 1 with the new value of a. }

procedure RRINFORM (G,R: LIST; VAR a,NF: LIST);
{Real root integral normal form. G reduced integral groebner basis of a nontrivial zerodimensional ideal, R is the set of reduced terms. NF is a list of entries (u,ut,ua,up) with: u is an element of R * R, RRREDTEST(G,u,_,ut), up is an integral polynomial and ua is an integer such that up/ua is the normal form of u wrt G. a is the lcm of all integers ua in NF. The elements of NF are sorted with respect to the actual termorder in decreasing order of the first entry. }

procedure RRISTRCONST (G,R: LIST; VAR U,a,beta: LIST);
{Real root integral structure constants. G reduced integral groebner basis of a nontrivial zerodimensional ideal, R the set of reduced terms. beta is the integral matrix of combined structure constants beta[u,v] wrt the basis a*R with u from U = R * R and v from R. U and the rows of the matrix beta are sorted with respect to the actual termorder in increasing order. }

procedure RRIVARMATRICES (G,R,U,beta: LIST; VAR al,L: LIST);
{Real root integral multiplication matrices of variables. G reduced integral groebner basis of a nontrivial zerodimensional ideal. R is the set of reduced terms in increasing order, U= R * R and beta is the set of combined structure constants wrt c*R for some nonzero integer c. al=(a(1),...,a(n)) is a list of integers and L is a list of entries of the form (1,M(i)) and M(i) is the matrix of multiplication with a(i)*X(i) wrt c*R. }

procedure RRIPOLMATRIX (R,h,f,fl: LIST; VAR Mh,L: LIST);
{Real root integral polynomial matrix. R is the set of reduced terms in increasing order, h is a polynomial of domain D, f is a positive integer, fl=(f(1),...,f(n)) is a list of positive integers, L contains nonempty lists L(i) of the form j(1),M(1),j(2),M(2),...,j(k),M(k) with 1=j(1)¡j(2)¡...¡j(k) and M(l) is the integral matrix of multiplikation with (f*f(i)*X(i))**j(l) for the variable X(i). L is extended with new calculated matrices. If d(i) is the maximal degree in the variable X(i) and d is the maximal total degree of the terms of the polynomial h, then Mh is the integral matrix of multiplication with (f**d)*(f(1)**d(1))*...*(f(1)**d(1)) * h. }

procedure RRIQUADFORM (R,U,beta,Mh: LIST): LIST;
{Real root integral quadratic form. R is the set of reduced terms in increasing order, U = R * R and beta is the set of integral combined structure constants wrt a*R for some nonzero integer a. Mh is the integral matrix of multiplication with c*h for some positive constant c represented columnwise. The matrix Q = (q(i,j)) with q(i,j) = tr(M(h)*M(v(i))*M(v(j)) and v(i),v(j) from a*R is computed. }

procedure RRICOUNT (G,H,v,tf: LIST): LIST;
{Real root integral count. G is an integral reduced groebner basis of a zerodimensional ideal I, H is a list of polynomials of length s. v is a vector of signs with length not greater than s. tf is the trace-flag. ZNL is a list of pairs (z,n) with n is an element of -1,0,+1**s and z ¿ 0 is the number of real zeroes of I wrt the sign condition n for the elements of H. ZNL is sorted wrt the invers lexicographical order of the n. If there does not exist any real zero or a zero satisfiing the sign condition v, then the empty list is returned. }

13.4 Real Root Univariate Arbitrary Domain

procedure RRUADPOLTOVEC (D,g,d: LIST): LIST;
{Real root univariate arbitrary domain polynomial to vector. g is an univariate polynomial of

domain D with degree less than d. If a(i) is the coefficient of X**i in g then the list (a(d-1),...,a(0)) is returned. }

procedure RRUADSTRCONST (D,f,h: LIST): LIST;
{Real root univariate arbitrary domain structure constants. f and h are univariate polynomials of domain D. f is monic with degree p ¿ 0. A matrix beta with entries beta[i,j] from D for 0 le i le p-1 and 0 le j le 3*p-3 is created, such that h * X**j = beta[0,j]+beta[1,j]*X+...+beta[p-1,j]X**(p-1) modulo f. beta is represented columnwise. }

procedure RRUADQUADFORM (beta: LIST): LIST;
{Real root univariate arbitrary domain quadratic form. beta is the set of structure constants as computed by RRUADSTRCONST. Let s(k) = tr(M(h)*M(X**k))=beta[0,k]+beta[1,k+1]+...+beta[p-1,k+p-1]. The matrix Q=(q(i,j)) with q(i,j) = s(i+j-2) is computed. }

procedure RRUADCOUNT (D,f,H,v,tf: LIST): LIST;
{Real root univariate arbirary domain count. f is a monic univariate polynomial of domain D with degree p ¿ 0. H is a list of univariate polynomials of length s. v is a vector of signs with length not greater than s. tf is the trace flag. ZNL is a list of pairs (z,n) with n is an element of -1,0,+1**s and z ¿ 0 is the number of real zeroes of f wrt the sign condition n for the elements of H. ZNL is sorted wrt the invers lexicographical order of the n. If there does not exist any real zero or a zero satisfiing the sign condition v, then the empty list is returned. }

13.5 Real Root Univariate Integral

procedure RRUIPOLTOVEC (g,d: LIST): LIST;
{Real root univariate integral polynomial to vector. g is an univariate integral polynomial with degree less than d. If a(i) is the coefficient of X**i in g then the list (a(d-1),...,a(0)) is returned. }

procedure RRUISTRCONST (f,h: LIST): LIST;
{Real root univariate integral structure constants. f and h are univariate integral polynomials. f has degree p ¿ 0. An integral matrix beta with entries beta[i,j] for 0 le i le p-1 and 0 le j le 3*p-3 is created, such that c*h*X**j = beta[0,j]+beta[1,j]*X+...+beta[p-1,j]X**(p-1) modulo f for some positive integer c. beta is represented columnwise. }

procedure RRUIQUADFORM (beta: LIST): LIST;
{Real root univariate integral quadratic form. beta is the set of structure constants as computed by RRUISTRCONST. Let s(k)=tr(M(c*h*X**k))=beta[0,k]+beta[1,k+1]+...+beta[p-1,k+p-1] for some positive constant c. The matrix Q = (q(i,j)) with q(i,j) = s(i+j-2) is computed. }

procedure RRUICOUNT (f,H,V,tf: LIST): LIST;
{Real root univariate integral count. f is an univariate integral polynomial with degree p ¿ 0. H is a list of univariate integral polynomials of length s. v is a vector of signs with length not greater than s. tr is the trace flag. ZNL is a list of pairs (z,n) with n is an element of -1,0,+1**s and z ¿ 0 is the number of real zeroes of f wrt the sign condition n for the elements of H. ZNL is sorted wrt the invers lexicographical order of the n. If there does not exist any real zero or a zero satisfiing the sign condition v, then the empty list is returned. }

Chapter 14

Logic Formulas

14.1 Maslog

procedure FORMKPOS (phi, pref: LIST;bbmkneg:PROCF1):LIST;
{formula make positive. phi is a formula; pref is a symbol of the set {ET, VEL, NON}; bbmkneg is a bb-procedure to negate a bb-formula; a formula equivalent to phi, which is relative positive, is returned. Relative positive means that negations are only in front of atomic formulas. pref is a switch that controls the substitution of the operators IMP, REP, EQUIV, XOR. If pref=NON then subformulas with IMP and REP are substituted, only if they are negated, EQUIV and XOR are not substituted. If pref=ET or pref=VEL then IMP and REP are substituted every time. If pref=ET (pref=VEL) then the outermost operator of the replacement for EQUIV and XOR is an ET (a VEL) operator. }

procedure FORMKDNF (phi: LIST; bbmkneg: PROCF1): LIST;
{formula make dnf. phi is a quantifier-free formula, bbmkneg is a bb-procedure to negate a bb-formula; a formula phi1 equivalent to phi is returned. phi1 is in strict dnf. }

procedure FORMKCNF (phi: LIST; bbmkneg: PROCF1): LIST;
{formula make cnf. phi is a quantifier-free formula, bbmkneg is a bb-procedure to negate a bb-formula; a formula phi1 equivalent to phi is returned. phi1 is in strict cnf. }

procedure FORMKPRENEX (phi,pref:LIST): LIST;
{formula make prenex. phi is a formula; pref is an element of {EXIST, FORALL}; a formula psi in prenex normal form is returned. phi must be a relative positive formula without additional operation symbols like IMP, REP, etc. All bound variables in phi must have different specifications (i.e. different names or different types). The only transformation which is used to calculate psi is the interchange of a junctor with a quantifier. The formula psi has the minimal number of blocks of quantifiers under all prenex formulas which are built using only the interchange of a junctor with a quantifier. The argument pref is only respected, if there are two equivalent formulas with the same optimal number of blocks of quantifiers. In this case the formula is returned which has a "pref"-quantifier as the outermost operation symbol.}

procedure FORMKPRENEXI (phi,pref:LIST): LIST;
{formula make prenex. phi is a formula; pref is an element of {EXIST, FORALL}; a formula psi in prenex normal form is returned. phi must be a relative positive formula without additional operation symbols like IMP, REP, etc. All bound variables in phi must have different specifications (i.e. different names or different types). The only transformation which is used to calculate psi is the interchange of a junctor with a quantifier. The formula psi has the minimal number of blocks of quantifiers under all prenex formulas which are built using only the interchange of a junctor

with a quantifier. The argument pref is only respected, if there are two equivalent formulas with the same optimal number of blocks of quantifiers. In this case the formula is returned which has a "pref"-quantifier as the innermost operation symbol.}

procedure FORMKPRENEX1 (phi: LIST): LIST;
{formula make prenex 1. phi is a relative positive formula; this procedure returns a list of maximal two objects of the form (psi,qb(psi)). In (psi,qb(psi)) psi is a formula in prenex normal form, and qb(psi) the number of blocks of quantifiers. If there are two objects in the returned list, then the outermost quantifier of the first formula is an exist quantifier and the outermost quantifier of the second formula is a forall quantifier. In the following comments such a list is called pnf-selection. }

procedure FORIMQB (phi: LIST):LIST;
{formula innermost quantifier block. phi is a formula in prenex normal form. If the outermost operator of phi is no quantifier then SIL is returned. Otherwise the type of the innermost quantifier block (either FOREX or FORALL) is returned. }

procedure FORSUBSTVAR (phi, old, new: LIST; bbsubstvar: PROCF3): LIST;
{formula substitute variable. phi is a formula; old and new are variables; a formula in which the variable old is substituted by the variable new is returned. }

procedure FORMKVD (formula:LIST; bbsubstvar:PROCF3;bblsvars:PROCF1): LIST;
{formula make variable names disjoint. formula is a formula, bbsubstvar is a bb-procedure to substitute variables in bb-formulas, bblsvars is a bb-procedure to list all variables in a bb-formula; FORMKVD returns a formula in which all bound variables of the same sort have different names. Only the minimal number of renamings are done to make the names different. }

procedure FORSMPL (phi: LIST; bbsmpl, bbmkneg: PROCF1): LIST;
{formula simplify. phi is a formula, simplifybb is a bb-procedure to simplify a bb-formula; bbmkneg is a bb-procedure to negate a bb-formula; a formula phi1 equivalent to phi is returned. The formula phi1 is simplified, that means the constants VERUM and FALSUM are eliminated, and nested operators are eliminated. (In this case the procedure takes advantage of associativity of ET and VEL, and the idempotenz of NON. }

procedure FORSIMPLIFY (phi: LIST; smart: PROCF1; bbsmpl, bbmkneg: PROCF1):LIST;
{formula simplify. phi is a formula, smart is a function to do smart simplification on a list of atomic formulas. simplifybb is a bb-procedure to simplify a bb-formula; bbmkneg is a bb-procedure to negate a bb-formula. A simplification of phi is returned. }

procedure FORSIMPLIFYP (phi,maxlevel: LIST;
smart: PROCF1; bbsmpl, bbmkneg: PROCF1):LIST; {formula simplify prune. phi is a formula, level, maxlevel are atoms, smart is a function to do smart simplification on a list of atomic formulas. simplifybb is a bb-procedure to simplify a bb-formula; bbmkneg is a bb-procedure to negate a bb-formula. maxlevel defines the number of levels that are simplified, 1 means only the top-level, zero means simplify the hole tree. A simplification of phi is returned. }

procedure FORPREPQE (phi: LIST; bbmkneg: PROCF1):LIST;
{formula prepare quantifier elimination. phi is a prenex formula; bbmkneg is a bb-procedure to negate a bb-formula; a formula psi equivalent to phi is returned. psi is built according to the following rules: If the innermost block of quantifiers is an exist-quantifier, then matrix(phi) is transformed in CNF and the innermost block of quantifiers is moved inside the conjunction. If the innermost quantifier is a forall-quantifier, then matrix(phi) is transformed in DNF and the innermost block of quantifiers is moved inside the disjunction.}

procedure FORELIMXOPS (phi :LIST; pref: LIST): LIST;
{formula eliminate extended operation symbols. phi is formula, pref is a symbol of the set {VEL, ET, NON}; FORELIMXOPS returns a formula phi1 equivalent to phi. If pref is NON then this function does nothing. Otherwise this function replaces all subterms of phi with the operators IMP, REP, EQUIV or XOR with terms with the operators VEL, ET and NON. There are two

different substitutions for EQUIV and XOR. If pref=ET (pref=VEL) then the outermost operator of the replacement terms for EQUIV, XOR is ET (VEL). }

procedure FORREPAFS (phi,rep:LIST):LIST;
{formula replace atomic formulas. phi is a formula. rep is a assoc list of the form (old1,new1,old2,new2,...), where old1,... and new1,.. are atomic formulas. Each occurence of oldi in phi is replaced with newi. }

procedure FORAPPLYAT (phi:LIST; dosomething:PROCF1): LIST;
{formula apply to atomic formular. phi is a formula; a formular in which all atomic formulas psi are substituted with dosomething(psi) is returned. }

procedure FORAPPLYATF2 (phi,param1:LIST; dosomething:PROCF2): LIST;
{formula apply to atomic formula f2. phi is a formula; param1 is an arbitrary list object, a formula in which all atomic formulas psi are substituted with dosomething(psi,param1) is returned. }

procedure FORCOUNTAF (phi:LIST): LIST;
{formula count atomic formulas. phi is a formula; The number of the atomic formulas in the formula phi is returned. }

procedure FORCONTBDVAR (phi:LIST; svar: LIST): BOOLEAN;
{formula contain bound variable. phi is a formula; var is a variable; FORCONTBDVAR returns true, if and only if phi contains var as a bound variable. }

procedure FORCONTVAR (phi:LIST; svar: LIST; bbcontvar: PROCFB2): BOOLEAN;
{formula contain variable. phi is a formula; var is a variable; bbcontvar is a procedure, which tests whether a bb-formula contains a variable or not; FORCONTVAR returns true, if and only if phi contains var as a free variable. }

14.2 Maslog Demonstration

procedure MLDMKATOM (rel,left,right:LIST):LIST;
{maslog demonstration make atomic formula. rel is the relation symbol (either EQU or NEQ), left and right are beta integers or variables. The formula (left rel right) is returned. }

procedure MLDTST (l: LIST):LIST;
{maslog demonstration test 1. l is a list; MLDTST returns 1 if l represents a formula otherwise 0. }

procedure MLDMKPOS (phi: LIST): LIST;
{maslog demonstration make positive. phi is a formula; a formula equivalent to phi, which is relative positive, is returned. Relative positive means that negations are only in front of atomic formulas. }

procedure MLDMKPOS1 (phi, pref: LIST): LIST;
{maslog demonstration make positive 1. phi is a formula; pref is a symbol of the set {ET, VEL, NON}; a formula equivalent to phi, which is relative positive, is returned. Relative positive means that negations are only in front of atomic formulas. pref is a switch that controls the substitution of the operators IMP, REP, EQUIV, XOR.}

procedure MLDMKDNF (phi: LIST): LIST;
{maslog demonstration make disjunctive normal form. phi is a formula; MLMKDNF returns a formula in strict disjunctive normal form which is equivalent to phi. }

procedure MLDMKCNF (phi: LIST): LIST;
{maslog demonstration make disjunctive normal form. phi is a formula; MLDMKCNF returns a formula in strict conjunctive normal form which is equivalent to phi. }

procedure MLDMKPRENEX (phi, pref: LIST): LIST;
{maslog demonstration make prenex. phi is a formula; pref is an element of {EXIST, FORALL};

a formula psi in prenex normal form is returned. phi must be a relative positive formula without additional operation symbols like IMP, REP, etc. All bound variables in phi must have different specifications (i.e. different names or different types). The only transformation which is used to calculate psi is the interchange of a junctor with a quantifier. The formula psi has the minimal number of blocks of quantifiers under all prenex formulas which are built using only the interchange of a junctor with an quantifier. The argument pref is only respected, if there are two equivalent formulas with the same optimal number of blocks of quantifiers. In this case the formula is returned which has a "'pref"'-quantifier as the outermost operation symbol.}

procedure MLDSUBSTVAR (phi,old,new:LIST):LIST;
{maslog demonstration substitute variables. phi is a formula; old and new are variables; a formula in which the variable old is substituted by the variable new is returned. }

procedure MLDMKVD (phi: LIST):LIST;
{maslog demonstration make variables disjoint. formula is a formula, this procedure returns a formula in which all bound variables of the same sort have different names. Only the minimal number of renamings are done to make the names different.}

procedure MLDSMPL (phi: LIST): LIST;
{maslog demonstration simplify. phi is a formula; a formula phi1 equivalent to phi is returned. The formula phi1 is simplified, that means the constants VERUM and FALSUM are eliminated, and nested operators are eliminated. (In this case the procedure takes advantage of associativity of ET and VEL, and the idempotenz of NON. }

procedure MLDPREPQE (phi: LIST):LIST;
{maslog demonstration prepare quantifier elimination. phi is a prenex formula; a formula psi equivalent to phi is returned. psi is built according to the following rules: If the innermost block of quantifiers is an exist-quantifier, then matrix(phi) is transformed in CNF and the innermost block of quantifiers is moved inside the conjunction. If the innermost quantifier is a forall-quantifier, then matrix(phi) is transformed in DNF and the innermost block of quantifiers is moved inside the disjunction.}

procedure MLDAPPLYAT (phi:LIST):LIST;
{maslog demonstration apply to atomic formulas. phi is a formula; the formula phi is returned and all atomic formulas are written in the outout stream. }

procedure MLDPPRT (phi: LIST);
{maslog demonstration pretty print. phi is a formula; this procedure writes the formula phi formated in the output stream. }

procedure MLDTEXW (phi: LIST);
{maslog demonstration tex write. phi is a formula; this procedure writes phi in tex style in the output stream. }

procedure MLDCONTVAR (phi,var:LIST):LIST;
{maslog demonstration contain variable. phi is a formula, var is a variable; 1 is returned, if phi contains var as an unbound variable, otherwise 0 is returned. }

procedure MLDCONTBDVAR (phi,var:LIST):LIST;
{maslog demonstration contains bound variable. phi is a formula, var is a variable; 1 is returned, if phi contains var as bound variable, otherwise 0 is returned. }

procedure MLDPREAD ():LIST;
{maslog demonstration prefix read. A formula is read from the input stream. }

procedure MLDIREAD ():LIST;
{maslog demonstration infix read. A formula is read from the input stream. }

14.3 Maslog Base

procedure FORGOP (phi: LIST): LIST;
{formula get operation. phi is a formula; the operation keyword of the formula phi is returned.}

procedure FORGARGS (phi: LIST): LIST;
{formula get arguments. phi is a formula; the arguments of the top level operation of the formula phi are returned. You must not apply FORGARGS to formulas without arguments! }

procedure FORGLVAR (lvar: LIST): LIST;
{formula get list of variables. lvar is an object that describes a list of variables; FORGLVAR returns a list of all variables in lvar. }

procedure FORPFOR (phi: LIST; VAR op: LIST; VAR args: LIST);
{formula parse formula. phi is a formula; FORPFOR returns in op the top level operation symbol of phi and in args the arguments of the operation op. }

procedure FORPARGS (phi: LIST; VAR first, red: LIST);
{formula parse arguments. phi is an object that describes a list of arguments of an operation; FORPARGS returns in first the 1st argument of the list and in red the list of all following arguments. }

procedure FORPUNOP (phi: LIST; VAR op, arg: LIST);
{formula parse unary operation. phi is a unary operation. The operator symbol op and the argument arg are returned. }

procedure FORPBINOP (phi: LIST; VAR op, first, second: LIST);
{formula parse binary operation. phi is a binary operation. The operator symbol op and the arguments first and second are returned. }

procedure FORPUNOPA (arglist: LIST; VAR arg: LIST);
{formula parse unary operation argument. arglist is a list containing the argument of a unary operator; FORPUNOPA returns in arg the arguments }

procedure FORPBINOPA (red: LIST; VAR first, second: LIST);
{formula parse binary operation argument. red is a list of the arguments of a binary operator; FORPBINOPA returns in first the 1st and in second the 2nd argument. }

procedure FORPQUANT (phi: LIST; VAR quant, vars, formula: LIST);
{formula parse quantifier. phi is a quantified formula; FORPQUANT returns in quant the quantifier of the formula phi, in vars the list of the bound variables (as an lvar object) and in formula the formula phi without quantifier. }

procedure FORPQUANTA (red: LIST; VAR lvar, formula: LIST);
{formula parse quantifier arguments. red is the reductum of a quantified formula (e.g returned from FORPFOR); FORPQUANTA returns in lvar a list of all quantified variables as an lvar object and in formula the formula without quantifier. }

procedure FORPLVAR (lvar: LIST; VAR varlist: LIST);
{formula parse list of variables. lvar is an object that describes a list of variables; the list of all variables in lvar is returned in varlist. }

procedure FORPVAR (var: LIST; VAR name, sort: LIST);
{formula parse variable. phi is a variable; FORPVAR returns in name the name and in sort the sort of var. }

procedure FORPVARA (red: LIST; VAR name, sort:LIST);
{formula parse variable arguments. red is the reductum of a variable (e.g. returned from FORPFOR); FORPVARA returns in name the name and in sort the sort of the variable. }

procedure FORMKFOR (op,args:LIST):LIST;
{formula make formula. op is a operation symbol; args a list of arguments; the formula

op(arg1,arg2,...) is returned, if args=(arg1,arg2,...). You can generate constants with this function, too. }

procedure FORMKCNST (cnst: LIST):LIST;
{formula make constant, i.e. a function with 0 arguments. cnst is a constant; the formula 'cnst()' is returned }

procedure FORMKUNOP (op,arg:LIST):LIST;
{formula make unary operation. op is a operation symbol, arg is a argument; the formula op(arg) is returned. }

procedure FORMKBINOP (op,arg1,arg2:LIST): LIST;
{formula make binary operation. op is the operation symbol, arg1, arg2 are the arguments; the formula op(arg1,arg2) is returned. }

procedure FORMKVAR (name,sort:LIST):LIST;
{formula make variable. name is the name, sort the sort of the variable; an object that describes the variable is returned. }

procedure FORMKLVAR (vars:LIST):LIST;
{formula make list of variables. vars is a list of the form (var1, var2,...); an lvar object that represents the list of variables in vars is returned. }

procedure FORMKQUANT (quant,vars,formula:LIST):LIST;
{formula make quantifier. quant is the quantifier symbol, vars is a list of variables, and formula the bound formula; the formula 'quant vars: formula' is returned. }

procedure FORTST (L: LIST; bbtst:PROCFB1):BOOLEAN;
{formula test. L is a list, bbtst a bb-procedure to test whether a list represents a bb-formula or not; FORTST returns TRUE, if and only if L represents a formula. }

procedure FORISVAR (L: LIST):BOOLEAN;
{formula is variable. L is a list; FORISVAR returns TRUE if and only if L represents a variable. (The type of the variable is not checked.) }

procedure FORISBOOLVAR (L: LIST):BOOLEAN;
{formula is boolean variable. L is a list; FORISBOOLVAR returns true if and only if L represents a boolean variable }

procedure FORISLVAR (L: LIST): BOOLEAN;
{formula is variable list. L is a list; FORISLVAR returns TRUE if and only if L represents an lvar-object. }

procedure FORISATOM (phi: LIST): BOOLEAN;
{formula is atomic formula. phi is a formula, FORISATOM returns true, if and only if phi is an atomic formula. An atomic formula is either VERUM or FALSUM or a bb-formula or a variable of type bool. }

procedure FORISBBFOR (phi: LIST):BOOLEAN;
{formula is black-box formula. phi is a formula. TRUE is returned iff phi is a black-box formula. }

procedure FORVTSTORE ():LIST;
{formula variable table store. The actual VARTAB is returned. }

procedure FORVTRESTORE (vt: LIST):LIST;
{formula variable table restore. Vt is a variable table. The actual VARTAB is set to the value of vt. The old value of VARTAB is returned. }

procedure FORVTENTER (sym: LIST):LIST;
{formula variable table enter. Sym is a CLIST. The symbol with print name sym is entered in the actual variable table, if no symbol with name sym exists. The unique identifier is returned. }

procedure FORVTGET (id: LIST): LIST;
{formula variable table get. The clist of the name of the symbol with identifier id is returned. If no symbol with identifier id exists, then SIL is returned. }

14.4 Maslog Input Output System

procedure FORPPRT (phi: LIST; bbpprt:PROCP1);
{formula pretty print. phi is a formula; bbpprt a bb-procedure to write a bb-formula to the output stream; phi is written formated to the output stream. }

procedure FORPPVAR (var: LIST);
{formula pretty print variable. var is a variable; var is formated written to the output stream. }

procedure FORPPLVAR (lvar: LIST);
{formula print lvar. L is an lvar object; lvar is formated written to the output stream. }

procedure FORTEXW (phi: LIST; bbtexw:PROCP1);
{formula tex write. phi is a formula; bbtexw a bb-procedure to write a bb-formula in tex style to the outputstream; this procedure writes a formula in tex style to the outputstream. }

procedure FORTEXWVAR (var: LIST);
{formula tex write variable. var is a variable; this procedure writes var in tex style to the output stream. }

procedure FORTEXWLVAR (lvar: LIST);
{tex write list of varaiables. lvar is an lvar object; this procedure writes all variables in lvar in tex style to the outputstream. }

procedure FORPREAD (bbread:PROCF0):LIST;
{formula prefix read. bbread is a bb-procedure to read a bb.formula from the input stream. FORPREAD reads a formula in prefix notation from the input stream and returns the read formula. }

procedure KEYREAD (): LIST;
{key read. A keyword or a symbol for a keyword is read. A keyword is a string of letters ore a string of special characters. The keyword or the symbol is also terminated by the characters "()[]". This procedure is based on the procedure SREAD1 from the module MASSYM2 }

procedure FORRDVAR ():LIST;
{formula read variable. A description of a variable is read from the input stream. }

procedure FORRDLVAR ():LIST;
{formula read list of variables. One variable or a list of variables are read from the input stream. A LVAR-object is returned. }

procedure FORIREAD (bbread:PROCF0):LIST;
{formula infix read. bbread is a bb-procedure to read a bb.formula from the input stream. This procedure reads a formula in infix notation from the input stream and returns the read formula. }

14.5 Polynomial Equation Base

procedure vlistflvar (lvar:LIST):LIST;
{variable list from lvar. lvar is a LVAR object. A variable list in the format of the module DIPC representing the same list of variables is returned. }

procedure lvarfvlist (vlist:LIST):LIST;
{lvar from variable list. vlist is a variable list in the format of the DIPC-module. A LVAR object

representing the same variable list is returned. }

procedure pqmkaf (rel:LIST;pol:LIST):LIST;
{polynomial equation simplification make atomic formula. rel is a relation, pol is a polynomial, the atomic formula (rel,id) is returned. }

procedure pqpaf (af:LIST; VAR rel,pol:LIST);
{polynomial equation simplification parse atomic formula. af is an atomic formula; the relation symbol of af is in rel returned; the polynomial of af is in id returned. }

procedure pqgrel (af:LIST):LIST;
{polynomial equation get relation symbol. af is an atomic formula. The relation symbol of af is returned. }

procedure pqgpol (af:LIST):LIST;
{polynomial equation get polynomial. af is an atomic formula. The polynomial of af is returned. }

procedure pqatom (phi:LIST):BOOLEAN;
{polynomial equation atomic formula. pqatom returns true iff phi has the structure of an atomic formula, i.e. phi is a list with two elements, and the first element of the list is an valid relation symbol. }

procedure pqprtaf (af: LIST);
{polynomial equation simplification print atomic formula. The atomic formula af is printed. }

procedure pqtexwaf (af: LIST);
{polynomia equation tex write atomic formula. The atomic formula af is written to the output stream. }

procedure pqnegaf (af: LIST):LIST;
{negate atomic formula. af is a atomic formula. The negation of af is returned. }

procedure pqsimplifyaf (af:LIST):LIST;
{polynomial equation simplify atomic formula. af is an atomic formula. af is simplified and af the result is returned. Only the relations between a constant polynomial (and zero) are evaluated. Be careful: Use only polynomials over an domain with a proper ADSIGN function. (For example: RN, but not RF.) }

procedure pqreadaf ():LIST;
{polynomial equation read atomic formula. An atomic formula is read from the input stream. The global variable DOMAIN must be set. Atomic fomulas have the following syntax: "¡dilp¿ ¡rel¿ ¡dip¿", where dip are distributive polynomials over an arbitrary domain with the variable list VALIS and rel is one of the sympols ¡,=,¿,¿=,¡¿,¡=,#,LES,EQU,GRE,LSQ,NEQ,GRQ,LEQ,GEQ. }

procedure InitBbfParser ();
{Initialize black-box formula parser. }

procedure pqsmart (phi:LIST):LIST;
{polynomial equation atomic formula smart simplification. phi is a conjunction or a disjunction over atomic formulas. All atomic formulas with identical left hand sides are contracted to one atomic formula. A conjunction or a disjunction over these contraced formulas is returned. }

procedure ContractVel (l:LIST):LIST;
{contract vel. l is a list of atomic formulas. These atomic formulas are interpreted as arguments of a disjunction. The relations of atomic formulas with equal left hand sides are contracted. }

procedure PQCRELOR (left,right:LIST):LIST;
{contract relations or. left and right are relations, the contraction of left an right are returned. (this procedure works correct, even if left=SIL.) }

procedure ContractEt (l:LIST):LIST;
{contract vel. l is a list of atomic formulas. These atomic formulas are interpreted as arguments of a disjunction. The relations of atomic formulas with equal identifiers are contracted. }

procedure PQCRELAND (left,right:LIST):LIST;
{contract relations or. left and right are relations, the contraction of left an right are returned. (this procedure works correct, even if left=SIL.) }

procedure PQMKDNF (phi:LIST):LIST;
{polynomial equation make disjunctive normal form. phi is a formula; MLMKDNF returns a formula in strict disjunctive normal form which is equivalent to phi. }

procedure PQMKCNF (phi:LIST):LIST;
{polynomial equation make disjunctive normal form. phi is a formula; a formula in strict conjunctive normal form which is equivalent to phi is returned. }

procedure PQSMPL (phi:LIST):LIST;
{polynomial equation simplify. phi is a formula; a simplification on phi is returned. }

procedure PQSIMPLIFY (phi:LIST):LIST;
{polynomial equation simplify. phi is a formula. A simplification of phi is returned. Following simplification steps are done: 1. VERUM and FALSUM are eliminated 2. assoziative simplification 3. idempotenz }

procedure PQSIMPLIFYP (phi,maxlevel:LIST):LIST;
{polynomial equation simplify. phi is a formula. A simplification of phi is returned. Following simplification steps are done: 1. VERUM and FALSUM are eliminated 2. assoziative simplification 3. idempotenz maxlevel is the number of levels that are simplified. }

procedure PQMKPOS (phi: LIST): LIST;
{polynomial equation make positive. phi is a formula; a equivalent positive formula is returned. }

procedure PQPPRT (phi:LIST);
{polynomial equation pretty print. phi is a formula; this procedure writes the formula phi formatted in the output stream. }

procedure PQTEXW (phi: LIST);
{polynomial equation tex write. The formula phi is printed in tex format in the output stream. }

procedure PQPREAD ():LIST;
{polynomial equation read. }

procedure PQIREAD ():LIST;
{polynomial equation infix read. }

procedure PQELIMXOPS (phi: LIST):LIST;
{polynomial equation eliminate extended operation symbols. phi is formula, pref is a symbol of the set {VEL, ET, NON}; FORELIMXOPS returns a formula phi1 equivalent to phi. If pref is NON then this function does nothing. Otherwise this function replaces all subterms of phi with the operators IMP, REP, EQUIV or XOR with terms with the operators VEL, ET and NON. There are two different substitutions for EQUIV and XOR. If pref=ET (pref=VEL) then the outermost operator of the replacement terms for EQUIV, XOR is ET (VEL). }

procedure PQELIMXOPS1 (phi,pref: LIST):LIST;
{polynomial equation eliminate extended operation symbols. phi is formula, pref is a symbol of the set {VEL, ET, NON}; FORELIMXOPS returns a formula phi1 equivalent to phi. If pref is NON then this function does nothing. Otherwise this function replaces all subterms of phi with the operators IMP, REP, EQUIV or XOR with terms with the operators VEL, ET and NON. There are two different substitutions for EQUIV and XOR. If pref=ET (pref=VEL) then the outermost operator of the replacement terms for EQUIV, XOR is ET (VEL). }

procedure PQMKPRENEX (phi,pref:LIST): LIST;
{polynomial equation make prenex. phi is a formula; pref is an element of {EXIST, FORALL}; a formula psi in prenex normal form is returned. phi must be a relative positive formula without additional operation symbols like IMP, REP, etc. All bound variables in phi must have different specifications (i.e. different names or different types). The only transformation which is used to calculate psi is the interchange of a junctor with a quantifier. The formula psi has the minimal number of blocks of quantifiers under all prenex formulas which are built using only the interchange of a junctor with a quantifier. The argument pref is only respected, if there are two equivalent formulas with the same optimal number of blocks of quantifiers. In this case the formula is returned which has a "pref"-quantifier as the outermost operation symbol.}

procedure PQPRING (R: LIST): LIST;
{polynomial equation polynomial ring. The global variables that describe the polynomial ring are set. The list R is of the following format: The first entry is the domain descriptor of the field, the second entry is the list of the variables, and the third entry is the term order. You can omit an entry of R by writing a -1 on the place of the entry. Not all entries must specified. The old parameters are returned. }

procedure PQPRINGWR ();
{polynomial equation polynomial ring write. The description of the polynomial ring is written in the output stream. }

procedure PQMKVD (phi:LIST): LIST;
{polynomial equation make variable names disjoint. }

14.6 Polynomial Equation Simplification

procedure PQSCNF (phi: LIST):LIST;
{polynomial equation simplification normal form. phi is an arbitrary quantifier-free formula. A equivalent formula in SCNF is returned. }

procedure PQSDNF (phi: LIST):LIST;
{polynomial equation simplification normal form. phi is an arbitrary quantifier-free formula. A equivalent formula in SDNF is returned. }

procedure PQCnfSimplify (nu:LIST):LIST;
{polynomial equation cnf based simplification. nu is an quantifier free formula. The formula nu is simplified. Consult the documentation for a description of the rules which are applied }

procedure PQDnfSimplify (nu:LIST):LIST;
{polynomial equation dnf based simplification. nu is an quantifier free formula. The formula nu is simplified. Consult the documentation for a description of the rules which are applied }

procedure SimplifyNf (nf: LIST):LIST;
{simplify normal form. nf is a formula in disjunctive normal form or conjunctive normal form. A simplification of nf is returned. Following rules are applied: Equal literals in clauses are contracted, atomic formulas with identical polynomials are contracted to TRUE or FALSE. Note: identical clauses are not contracted. This happens in the GetDataXXX Procedures. In this procedure equal implications are contracted. }

procedure IdealMember (G,p:LIST):BOOLEAN;
{ideal membership test. G is groebner basis. Iff p in ID(g) then 1 is returned otherwise 0. }

procedure RadicalMember (G,p:LIST):BOOLEAN;
{radical membership test. G is a groebner basis. Iff p in RAD(G) then 1 is returned otherwise 0. The new variable Rw is introduced. }

procedure DIPADGB (P:LIST):LIST;
{distributive polynomial arbitrary domain groebner basis. P is a list of polynomials (over the

ring of integers). A groebner basis of P is returned. This procedure is at moment only a dummy procedure. It should calculate a groebner basis in respect to the coefficient ring of the polynomials. }

procedure DIPADNF (P,S:LIST):LIST;
{distributive polynomial arbitrary domain normal form. P is a list of polynomials. The normal form of the polynomial S w.r.t. P is returned. This procedure is at moment only a dummy. Is should work on Polynomials over arbitrary fields and rings. }

procedure DILADNF (P,S:LIST):LIST;
{distributive polynomial list arbitrary domain normal form. P is a list of polynomials. S is a list of polynomials. The list of normal forms of the polynomials of S w.r.t. P is returned. This procedure is at moment only a dummy. Is should work on Polynomials over arbitrary fields and rings. }

procedure DIPADGBext (gb,pols:LIST):LIST;
{distributive polynomial arbitrary domain groebner basis extension. gb is a groebner basis, pols is a list of polynomials. A groebner basis of the ideal basis gb join pols is calculated and returned. This procedure is at moment only a dummy. }

procedure DIPADGBunion (gb1,gb2:LIST):LIST;
{distributive polynomial arbitrary domain groebner basis union. gb1 and gb2 are groebner basis. A groebner basis of the ideal basis gb1 join gb2 is calculated and returned. This procedure is at moment only a dummy. }

procedure DIPADIRSET (P:LIST):LIST;
{distributive polynomial arbitrary domain irreducible set. P is a list of polynomials. A set PP of polynomials is returned. PP is the result of reducing each element p modulo P - p until no further reductions are possible. }

procedure DIPADGBRED (gb: LIST):LIST;
{distributive polynomial groebner basis reduction. gb is a groebner basis of distributive polynomials over an arbitrary domain. The unique reduced and ordered groebner basis to gp is returned. }

procedure PQOPT (O:LIST):LIST;
{polynomial equation options. The options of the PQ-System are set. The list O is of the following format: The first entry is the trace level of the system, the second entry determines the method for the radical member ship test, the third entry is a list of powers for the pseudo radical member ship test, the fourth entry controls the replacement of the premises of the implications, and the fifth entry controls the You can omit an entry of O by writing a -1 on the place of the entry. You need not specify all entries. The old parameters are returned. }

procedure PQOPTWR ();
{polynomial options write. The options of the PQ-System are printed in the output stream. }

procedure PQDEMO ();
{Demonstration for this package. }

procedure DLSWRITE (S:ARRAY OF CHAR;b:LIST);
{debug level SWRITE. The string S is in dependency of the debug level written to the output stream. In contrast to MAS a blank line is added. (analogous to UWRITE and UWRIT1) }

procedure SETADD (a,b:LIST):LIST;
{set add element. If the element elem is not in the set set, then elem join set is returned else set is returned. }

procedure rabinowitsch (p:LIST):LIST;
{rabinowitsch. p is a non zero polynomial. The polynomial 1-Zp is returned. Z denotes the variable with the exponent vector (0,...,0,1). }

14.7 Real Quantifier Elimination with Parametric Real Root Count.

procedure RQEQE (phi:LIST):LIST;
{real quantifier elimination quantifier elimination. phi is a formula. A formula psi equivalent to phi is returned. All quantifiers must bound different variables. No variable is allowed to occur free and bound. An automatic renaming of variables is not done. Atomic formulas must be truth values or must have the form "(rel term)", where rel is an relation and term is distributive polynomial over the domain INT. All atomic formulas of the latter form of atomic formulas must contains polynomials in the same polynomial ring, which is determined by the global variables VALIS, EVORD , and DOMAIN. In the normal case psi contains no quantifier. See the documentation of the options of the RQE-SYSTEM for more informations. The global variables VALIS, EVORD and DOMAIN must be set appropriately. The options for the CGB-SYSTEM must be set appropriately. VALIS must contain the variable list of the polynomials used in the atomic formulas. EVORD must contain the term order of the polynomials used in the atomic formulas. DOMAIN must contain the domain descriptor for the PQ-SYSTEM. Only the domain INT is valid. The term orders of the CGB-SYSTEM and the variable EVORD must be set compatible. All term orders should be equal. Tracing of intermediate output, conditions to the output formula and other things are controlled by the RqeOpt variable. This procedure calls the CGB-SYSTEM. Use the options of this system for controlling the computation of an Groebner system. }

procedure RQEOPTWR ();
{real quantifier elimination option write. The actual options are written to the output stream. }

procedure RQEOPTSET (opt:LIST):LIST;
{real quantifier elimination options set. opt is a list of options. The first element of opt is the trace level. The second element of opt is a flag. If this flag is true, then partial elimination of zero dimensional ideals are done. Otherwise full quantifier elimination is done. The old option list is returned. }

14.8 Type Formula

procedure tfmkaf (rel:LIST;idl:LIST):LIST;
{type formula make atomic formula. rel is a relation, idl is a list of atoms, the atomic formula (rel,idl) is returned. }

procedure tfpaf (af:LIST; VAR rel,idl:LIST);
{type formula parse atomic formula. af is an atomic formula; the relation symbol of af is in rel returned; the list of identifiers is in id returned. }

procedure tfgrel (af:LIST):LIST;
{type formula get relation symbol. af is an atomic formula. The relation symbol of af is returned. }

procedure TFPPRT (phi:LIST);
{type formula pretty print. phi is a tf-formula; this procedure writes the formula phi formatted in the output stream. }

procedure TFIREAD ():LIST;
{type formula infix read. A type formula is read from the input stream. }

procedure TfUseDb ();
{type formula use data base. The global variable TypeFormulaProc is set. Type formulas are computed only if they are not stored in the data base. }

procedure TfComputeTf ();
{type formula compute type formulas. The global variable TypeFormulaProc is set. The data

base of type formulas is not used. }

procedure TfTypeFormula (deg: LIST):LIST;
{type formula type formula. A type formula for polynomials of degree deg ¿ 0 is returned. This is the interface for other modules. }

procedure ComputeTypeFormula (deg: LIST):LIST;
{compute type formula. deg is an atom ¿ 0. A type formula for polynomials of degree deg is returned. }

procedure UseDb (deg: LIST): LIST;
{Use data base. A type formula for polynomials of degree deg ¿ 0 is returned. If the type formula is already stored in the data base, this formula is returned. Otherwise the type formula is computed, stored in the data base and returned. Type formulas are stored in files with the name TF.d.db, where d is the degree of the type formula. }

procedure TFGENJ (deg: LIST):LIST;
{type formula generate with joker argument. deg is an even atom greater than 1. computed. The computed formula is returned. A strict type formula for polynomials of degree deg is computed. The joker argument is used in order to reduce the size of the type formula. }

procedure TfRecBasis (deg:LIST):LIST;
{type formula recursion basis. deg is an atom ¡ 6. A type formula for polynomials with degree deg is returned. }

procedure TfCtj (deg: LIST):LIST;
{type formula coefficient tuples with joker argument. deg is an even atom. All good coefficients tuples with of polynomials of degree deg are computed. }

procedure TfShiftVars (phi,offset: LIST):LIST;
{type formula shift variables. phi is a type formula, offset is an atom. All identifiers are shifted at offset positions to the right. }

procedure tfshiftaf (phi,offset:LIST):LIST;
{type formula shift atomic formula. phi is an atomic formula, offset is an atom. all identifiers in phi are shifted. }

procedure TFFTUPLE (tup:LIST):LIST;
{type formula from coefficient tuple with joker entries. tup is a coefficient tuple. A formula in tf-format is returned. The formula is true, iff the instantiation of the variables suffice the relation conditions of the characteristic coefficients of tup. }

procedure Class2Sym (class: TfClass):LIST;
{classification to symbol. class is a constant of type class. The corresponding mas symbol is returned. }

procedure Sym2Class (class: LIST):TfClass;
{symbol to classification. class is a mas-symbol the corresponding classification is returned. }

procedure InitClassSyms ();
{Initialize classification symbols. The mas symbols representing classification constants are initialized. }

procedure TfClassify (tup:LIST):TfClass;
{type formula classify coefficient tuple. tup is a coefficient tuple. The classification of tup is returned. Let f the polynomial which is represented by tup. If tau(f)=0, then Tfgood is returned, if tau(f)¡0, then Tfdontcare is returned, if tau(f)¿0, then Tfbad is returned. Is the sum of the upper bounds of the real zeroes do not equal the degree of f, then TfImpossible is returned. }

procedure TfClassifyI (tup:LIST):LIST;
{type formula classify coefficient tuple interpreter version. tup is a coefficient tuple. The classification of tup is returned. Let f the polynomial which is represented by tup. If tau(f)=0, then TfGood is returned, if tau(f)¡0, then TfDontCare is returned, if tau(f)¿0, then TfBad is returned.

Is the sum of the upper bounds of the real zeroes not equal to the degree of f, them TfImpossible is returned. The classification is returned as an symbol, representing the classification.}

procedure TFGENI (deg: LIST; class: LIST): LIST;
{TFGEN interpreter version. Like TFGEN but the classification parameter is a MAS symbol. }

procedure TFGEN (deg: LIST; class: TfClass): LIST;
{type formula generate. deg is an atom ¿ 1. A tf-formula is returned. This formula is true, iff a tuple t with t_0¡¿0 has the classification class. }

procedure PatternAStart (deg:LIST; VAR start, pattern:LIST);
{pattern and start. A pattern and a start value for the computation of characteristic coefficient tuples is computed. Let (c_0,...,c_deg) the start value and (p_0,...,p_deg) the pattern. Then the following holds. If d MOD 4 = 0, then c_0=1 p_0=0. If d MOD 4 = 1, then c_0=0 and c_1=1 and p_0=p_1=0. If d MOD 4 = 2, then c_0=-1 and p_0=0. If d MOD 4 = 3, then c_0=0 and c_1 = -1 and p_0=p_1=0. c_deg=1. }

procedure TfNextTuple (last,pattern:LIST):LIST;
{type formula next tuple. last is a list with n elements of the set {0,-1,1}. pattern is a list with n elements of the set {0,1}. The lexicographic tuple which differs only an those positions that are 1 in list pattern is returned. The constant SIL is returned, if no lexicographic tuple greater last exists. The result is generated non-constructive from last. }

procedure NextRel (e: LIST):LIST;
{next relation. e is an code for a relation. The code of the next relation (in order ¡, =, ¿) is returned. }

procedure TfCount (deg: LIST):LIST;
{type formula count. deg is an atom. Some characteristic coefficient tuples (c_0,...,c_d) in $\{-1,0,1\}\hat{}(d+1)$ are generated. The following holds. If d MOD 4 = 0, then c_0=1. If d MOD 4 = 1, then c_0=0 and c_1=1. If d MOD 4 = 2, then c_0=-1 If d MOD 4 = 3, then c_0=0 and c_1 = -1. c_deg=1. A list (Good,Bad,DontCare,Impossible) is returned. Each element is the number of coefficient tuples with the corresponding classification. }

procedure TfCount1 (start, pattern: LIST):LIST;
{type formula count 1. start is a list (s_0,...s_d) with elements of the set {-1,0,1}, pattern is a list (p_0,...,p_d) with elements of {0,1}. The set T := t in -1,0,1**d+1 — t ¿= s , t_i=s_i, if p_i=0 is computed. A list (Good,Bad,DontCare,Impossible) is returned. Each element is the number of coefficient tuples with the corresponding classification. }

procedure TfZeroes0 (tup: LIST):LIST;
{type formula zeroes 0. The numbers of the leading zeroes is returned. This is the number of zeroes at the origin of the polynomial SUM tup_i X**i. For the tuple tup containing only zeroes and for the empty tuple the constant SIL is returned. }

procedure TfZeroes (tup: LIST; VAR CZeroes, ZeroesM, Zeroes0, ZeroesP: LIST);
{type formula zeroes. tup is a coefficient tuple of a univariate polynomial f. A upper bound of the real zeroes of f, which are greater than zero (less than zero) is returned in ZeroesP (ZeroeseM). The number of zeroes at the origin are returned in Zeroes0. If f has only real zeroes then the numbers of zeroes are exactly. The computation of the numbers are done using Descartes rule of signs. The number of all complex zeroes (=deg(f)=length(tup)-1) is returned in CZeroes. If tup is a list containing only zeroes, then CZeroes is SIL, and all other variables are undefined. }

procedure TfZeroesI (tup: LIST):LIST;
{TfZeroes interpreter version. This procedure returns the 4 variable parameters of the procedure TfZeroes as a list containing 4 elements. }

procedure TfSignChs (tup: LIST): LIST;
{type formula sign changes. tup is a list with elements of the set {-1,0,1}. The number of sign changes of tup is returned. }

Chapter 15

Involutive Bases

15.1 Arbitrary domain extra definition module

procedure ADPCP (A: LIST): LIST;
{Arbitrary Domain polynomial content and primitive part. A is an arbitrary domain polynomial, The result is the positive primitive part of A. }

procedure ADPNEG (A: LIST): LIST;
{Arbitrary domain polynomial negative. Input: an arbritrary domain polynomial A, Output: -A }

procedure ADPIQ (A,b: LIST): LIST;
{Arbitrary domain polynomial integer quotient. Input: A is an arbitrary domain polynomial, b is a nonzero integer, and b divides any coefficient of A. Output: C=A/b.}

procedure ADLGinH (H, G: LIST): BOOLEAN;
{Arbitrary domain polynomial list G in H. Input: H is a list of lists of arbitrary domain polynomials, G is a list of arbitrary domain polynomials. Output: TRUE iff ex. h in H s.t. G = h, FALSE else. }

procedure ADLGeqH (H, G: LIST): BOOLEAN;
{Arbitrary domain polynomial list G equal H. Input: H and G are lists of arbitrary domain polynomials, Ouput: TRUE iff H=G, FALSE else. }

procedure ADPFeqG (F, G: LIST): BOOLEAN;
{Arbitrary domain polynomial F equal G. Input: arbitrary domain polynomials F and G, Ouput: TRUE iff g = h, FALSE else. }

procedure ADIredG (I,G: LIST): LIST;
{Arbitrary domain polynomial set I reducible modulo G. Input: arbitrary domain polynomial sets I and G. Output: 0 iff all i in I are reducible modulo G to zero, a reduced polynomial p else }

procedure ADGJredI (G,I: LIST): LIST;
{Arbitrary domain polynomial G Janet-reducible modulo I. Input: arbitrary domain polynomial sets G and I. Output: 0 iff all g in G are Janet-reducible modulo I to zero, a reduced polynomial p else }

procedure IBeqGB (G,I: LIST): LIST;
{Inovlutive Base equal Groebner Base. Input: Groebner Base G and involutive Base I, Output: 0 iff Id(G) = Id(I), a reduced polynomial p else }

15.2 DIP Common Polynomial System in the sense of Janet.

procedure ADDTDG (deg, P: LIST): LIST;
{Add total degree. Input: polynomial degree deg and polynomial P in distributive list representation. Output: polynom P with first list entry now total degree of the leading exponent vector. }

procedure ADVTDG (P: LIST; VAR p, PP: LIST);
{Advance total degree. Input: polynom P in distributive list representation. The first entry of the list is the total degree of the leading exponent vector. Output: p - the first entry of the list; PP - the polynom without the first entry }

procedure DILEBBS (A: LIST);
{Distributive List Extended Bubble Sort. Sort a list of polynoms in decreasing order of the total degree of the leading exponent. The total degree must be the first entry of each polynom. Input: A is a list of polynoms. A is changed }

procedure DILBBS (A: LIST);
{Distributive List Bubble Sort. Sort a list of polynoms in decreasing order of the total degree of the leading exponent. Input: A is a list of polynoms. A is changed }

procedure DILEP2P (P: LIST): LIST;
{Distributive polynom list extended polynom to polynom. Input: P - a list of extended polynoms. Output: a list of polynoms whithout the first entry. }

procedure DILATDG (P: LIST): LIST;
{Distributive polynom list add total degree. P is a list of distributive polynomials. The result is a list of distributive polynoms with total degree of the leading term as first entry of each polynomial. }

procedure DILTDG (A: LIST): LIST;
{Distributive polynomial list total degree Input: A is a list of distributive polynomials, Output tdg: the total degree of A }

procedure DIPCLP (P: LIST): LIST;
{Distributiv Polynomial Class of Polynomial. Input: P is a polynomial, Output: t is the index of the lowest variable of the leading term of P, t=0 if P is Empty }

procedure DIPCLT (P: LIST): LIST;
{Distributiv Polynomial Class of Term. Input: P is a term, Output: t is the index of the lowest variable in P, t=0 if P is empty }

procedure DIPFIRST (P: LIST; extended: BOOLEAN; VAR pp, PP: LIST);
{Distributive polynomial first polynomial, P is a list of polynomials, pp is the first polynomial and PP is the reductum of P.}

procedure DIPSSM (P: LIST; extended: BOOLEAN; VAR pp, PP: LIST);
{Distributive polynomial sort and select minimal. Input: P - a list of polynoms, extended is TRUE iff the first entry of each polynimial in P is the total degree of the leading exponent vector. Output: pp - the minimal polynom w.r.t. the admissible term order. PP - sorted list of P without pp. P is changed }

procedure DILCAN (P: LIST; F: PROCF1): LIST;
{Distributive Polynomial Cancel. P is a list of distributive polynomials. F is the cancel-function. Output is a list of distributive polynomials or an empty list if all polynomials in A equal 0. The coefficients of each polynomial are canceld down by F. }

procedure DIILPP (P: LIST): LIST;
{Distributive integral polynomial list primitive part. P is a list of distributive integral polynoms. The result is the positive primitive part of each polynomial in P. The list-order is reversed. }

procedure DIRPMV (A,B: LIST): LIST;
{Distributiv Polynomial multiplication with a variable. Input: A is the polynomial, B is an exponent vector. Output: A*B }

procedure EVDIF2 (U,V: LIST): LIST;
{Exponent vector difference. Input: U=(u1, ...,ur), V=(v1, ...,vr) are exponent vectors of length r. Output: W=(w1, ...,wr) is the componentwise difference of U and V. Unlike procedure EVDIV this procedure returns 0 and not (0...0) if U=V }

procedure EVMTJ (U,V: LIST): LIST;
{Exponent vector multiple test in the sense of Janet. Input: U=(u1, ...,ur), V=(v1, ...,vr) are exponent vectors of length r. Output: t=1 if U is a multiple in the sense of Janet of V, t=0 else. }

procedure DIPNML (G: LIST): LIST;
{Distributive polynomial nonmultiple variable list. Compute for a polynom G the List of non-multiplicative variables. Input: G is a polynomial. Output: a list of non-multiplicative variables for the leading term of G. }

procedure DIPPGL2 (F, V, LL: LIST): LIST;
{Distributive polynomial prolongation list. Input: F - polynomial which first entry is the total degree of the leading term; V - list of variables; LL - number of different variables in F. Output: PP - List of prolongations of F with non-multiplicative variables for F from V. }

procedure DIPPGL3 (F, V, LL: LIST): LIST;
{Distributive polynom prolongation list. Input: F - polynomial; V - list of variables; LL - number of variables in F. Output: PP - List of prolongations of F with non-multiplicative variables for F from V. }

procedure DIPPGL (V: LIST): LIST;
{Distributive polynomial prolongation list. Input: V is an arbitrary domain polynomial. Output: List of prolongations of V with nonmultiplicative variables for V.}

procedure DIPVL (V: LIST): LIST;
{Distributive Polynomial List of Variables. Input: a polynomial V. Output: list of variables with class ¿ 1. }

15.3 DIP Decompositional Involutive Bases

procedure SetDCIBopt (options: LIST);
{Set decompositional involutive base options. Input: a list of max. 4 options in the order: TraceLevel, DecomProc, VarOrd, Depth of tree. }

procedure SetDCIBTraceLevel (TL: INTEGER);
{Set Decompositional involutive base Trace Level. Input: an integer 0¡= TL ¡= 3, 0: default, no output, ¿0: output of time, ¿1: output of messages about tree of computation, ¿2: detailed messages about tree of computation. }

procedure SetDCIBDecomp (DCP: INTEGER);
{Set decompositional involutive base decomposition. Set the procedure which is used for polynomial decomposition. 1: complete factorisation 2: squarefree decomposition }

procedure SetDCIBVarOrdOpt (VOO: INTEGER);
{Set decompositional involutive base variable order option. VOO is an integer with meaning: 0: do not optimize, 1: optimize at factorization }

procedure SetDCIBdepth (d: INTEGER);
{Set decompositional involutive base depth of tree. Input: an integer with ¡0: unrestricted growth of tree 0: no computation possible, ¿0: depth of tree is restricted through d }

procedure InvolutiveBases (G, V: LIST): LIST;
{Involutive Bases. G is a list of polynomials in distributive representation over an arbitrary domain, returns a list (IB1,...,IBk) of involutive bases, where Z(G) = Z(IB1) v ... v Z(IBk). }

procedure DILNFJ (H,G: LIST): INTEGER;
{Distributive Polynomial List normalform in the sense of Janet. H,G are distributive polynomial lists, returns 0 if each polynomial in H is Janet-reducible to 0 modulo G, 1 else. }

procedure IBLWR (PP,V: LIST);
{Involutive bases list write. PP is a list of involutive bases in distributive representation. V is a variable list }

procedure DecCounter (VAR counter: LIST; VAR length_of_counter: INTEGER);
{Decrement counter. counter is a list of integers, the first element of counter is removed }

procedure IncCounter (VAR counter: LIST; VAR loc: INTEGER; add: INTEGER);
{Increment counter. Increment the first entry of the counter list or append a new element. Input: counter: a list of integers, add = 0: append a new element and increment counterlength loc, or add¿0: increment the first element by add }

procedure CounterWR (counter: LIST);
{Counter Write. write the given list counter as the number of a reached node }

15.4 DIP Common Polynomial System in the sense of Janet.

procedure ADCAN (P: LIST): LIST;
{Arbitrary Polynomial Cancel. P is an polynomial of arbritrary domain, P is canceld down with ADPCP iff the domain of P is INT and with DIPMOC else }

procedure ADPNFJ (G,P: LIST; VAR h: LIST; VAR reduced: BOOLEAN);
{Arbitrary domain polynomial normalform in the sense of Janet. G is a list of polynomials in distributive representation over an arbitrary domain, P is a polynomial as above, returns a polynomial h such that P is Janet-reducible to h modulo G and h is in Janet-normalform w.r.t. G, the flag "reduced" is set TRUE iff a Janet-reduction took place }

procedure DIPNFJ (P,S: LIST; VAR h: LIST; VAR reduced: BOOLEAN);
{Distributive polynomial normal form in the sense of Janet. P is a list of non zero polynomials in distributive representation in r variables. S is a distributive polynomial. The result is a polynomial h such that S is reducible to h modulo P in the sense of Janet and h is in Janet-normalform with respect to P, "reduced" is set TRUE iff a Janet-reduction took place. }

procedure DIPINFJ (P,S: LIST; VAR h: LIST; VAR reduced: BOOLEAN);
{Integral Distributive polynomial normal form in the sense of Janet. P is a list of non zero polynomials in distributive representation in r variables. S is a distributive polynomial. h is a polynomial such that S is reducible to h modulo P in the sense of Janet and h is in normalform with respect to P, reduced is set TRUE iff a Janet-reduction took place. }

procedure DILISJ (P: LIST; VAR PP: LIST; VAR reduced: BOOLEAN);
{Distributive polynomial list irreducible set in the sense of Janet. P is a list of distributive polynomials, PP is the result of reducing each polynomial p in P modulo P-(p) in the sense of Janet until no further reductions are possible. reduced is set TRUE if a Janet-reduction took place. }

procedure DIPIRLJ (P: LIST; VAR F: LIST; VAR reduced: BOOLEAN);
{distributive polynomial interreduced list in the sense of Janet. P is a list of polynomials in distributive representation over an arbitrary domain, The result is a Janet-interreduced list of polynoms F, reduced is set TRUE iff a reduction took place. }

procedure DIPCOM (F: LIST): LIST;
{Distributive polynomial complete. Subalgorithm for computing Invbase. Input: Distributive polynomial list F. Output: G: complete system, such that Ideal(G) = Ideal(F). }

procedure DIPIB2 (F: LIST): LIST;
{Distributive polynomial involutive basis. Mainalgorithm for computing Invbase. Input: Distributive polynomial list F. Output: G: involutive system, such that Ideal(G) = Ideal(F). }

procedure DIPIB3 (F: LIST): LIST;
{Distributive polynom involutive base. Algorithm for computing the involutive Base for a given polynomial list F. Input: Distributiv Polynomial List F. Output: Equivalent involutive system G.}

procedure ADEPNFJ (G,P: LIST; VAR h: LIST; VAR reduced: BOOLEAN);
{Arbitrary domain extended polynomial normalform in the sense of Janet. G is a list of polynomials in distributive representation over an arbitrary domain, P is a polynomial as above, returns a polynomial h such that P is Janet-reducible to h modulo G and h is in Janet-normalform w.r.t. G, the flag "reduced" is set True iff a Janet-reduction took place }

procedure DIPENFJ (P,S: LIST; VAR R: LIST; VAR c: BOOLEAN);
{Distributive extended polynomial normal form in the sense of Janet. P is a list of non zero extended polynomials in distributive representation in r variables. S is a distributive extended polynomial. R is an extended polynomial such that S is reducible to R modulo P in the sense of Janet and R is in normalform with respect to P, c is set TRUE iff a Janet-reduction took place. }

procedure DIPEINFJ (P,S: LIST; VAR R: LIST; VAR c: BOOLEAN);
{Integral distributive extended polynomial normal form in the sense of Janet. P is a list of non zero extended polynomials in distributive representation in r variables. S is a distributive extended polynomial. R is a polynomial such that S is reducible to R modulo P in the sense of Janet and R is in normalform with respect to P. }

procedure IsInvolutive (G: LIST): BOOLEAN;
{Is involutive. Test wether G is involutive, G is a list of distributive polynomials, Returns TRUE iff G is involutive, FALSE else }

procedure ADEPmark (g, G: LIST): LIST;
{Arbitrary domain extended polynomial mark. g is an extended polynomial, G is a list of extended polynomials; the first entry of g is set to 0 and G is set to G cup g }

procedure DILISJ2 (P: LIST): LIST;
{Distributive polynomial list irreducible set. P is a list of distributive polynomials, PP is the result of reducing each p element of P modulo P-(p) in the sense of Janet until no further reductions are possible. }

procedure ADEPtriple (PS, term: LIST): LIST;
{Arbitrary domain extende polynomial triple. Input: PS - a list of polynomials, term - term to use as head term or SIL. Output: a list of extended polynomials }

procedure ADEPuntriple (PS: LIST): LIST;
{Arbitrary domain extended polynomial untriple. Input: PS - an extended polynomial list. Output: a polynomial list. }

procedure ADEPCrumble (P: LIST; VAR deg, pol, ht: LIST);
{Arbitrary domain extended polynomial crumble. Input: an arbritraty domain extended polynomial, Output: the components of the extended polynomial: deg - the total degree of the leading term, pol - the polynomial and if exists ht - the head term of the polynomial and 0 else. }

procedure ADEPCompose (deg, pol, ht: LIST): LIST;
{Arbitrary domain extended polynomial compose. Input: the components to build an extended polynomial: deg - the total degree of the polynomial, pol - the polynomial and ht - the head

term of the polynomial or 0 if ht should not be element of the extended polynomial. Output: an extended polynomial. }

procedure ADEPdegree (P: LIST): LIST;
{Arbitrary domain extended polynomial degree. Input: P - an arbitrary domain extended polynomial. Ouput: the degree of the head term of P, that is the first entry from the extended polynomial }

procedure ADEPpolynomial (P: LIST): LIST;
{Arbitrary domain extended polynomial polynomial. Input: P - an arbitrary domain extended polynomial. Output: the polynomial entry of the extended polynomial, that is the second entry from the extended polynomial. }

procedure ADEPheadterm (P: LIST): LIST;
{Arbitrary domain extended plynomial head-term. Input: P - an arbitrary domain extended polynomial. Output: the head term entry of the extended polynomial. That is the third entry in the extended polynomial list. The third list entry must not be equal to the head-term of the polynomial entry of the extended polynomial list. }

procedure ADEPleadingterm (P: LIST): LIST;
{Arbitrary polynomial leading term. Input: P - an arbitrary domain extended polynomial. Output: the head term of the polynomial in the extended polynomial list, that is the first element of the second list entry. }

procedure IBcrit (NM, G: LIST): LIST;
{Inovlutive Base criterium. Input: NM - a list of prolongated extended polynomials, G - a list of extended polynomials. Output: NM minus the polynomials from NM which are reducible to 0 by one of the criteriums, or which are reducible to 0 modulo G. }

procedure DIPIB (F: LIST): LIST;
{Distributive polynomial involutive basis. Algorithm for computing the involutive Base for a given F. Input: Distributiv Polynomial List F. Output: Equivalent involutive system G.}

procedure ADPNFJS (G1,G2,G3,G4,P: LIST; VAR h: LIST; VAR reduced: BOOLEAN);
{Arbitrary domain polynomial normalform in the sense of Janet modulo a set of set of polynomials. G1-G4 are lists of polynomials in distributive representation over an arbitrary domain, P is a polynomial. The result is a polynomial h such that P is Janet-reducible to h modulo the union of G1-G4 and h is in Janet-Normalform w.r.t. G1-G4, the flag "reduced" is set TRUE iff a Janet-reduction took place }

procedure DIPNFJS (P1,P2,P3,P4,S: LIST; VAR h: LIST; VAR reduced: BOOLEAN);
{Distributive polynomial normal form in the sense of Janet modulo a set of sets of polynomials. P1-P4 are lists of non zero polynomials in distributive representation in r variables. S is a distributive polynomial. R is a polynomial such that S is reducible to R modulo P in the sense of Janet and R is in normalform with respect to P, reduced is set TRUE iff a reduction took place. }

procedure DIPINFJS ;
{Integral Distributive polynomial normal form in the sense of Janet modulo a set of sets of polynomials. P1-P4 are lists of non zero polynomials in distributive representation in r variables. S is a distributive polynomial. h is a polynomial such that S is reducible to h modulo P in the sense of Janet and h is in normalform with respect to P, reduced is set TRUE iff a Janet-reduction took place. }

procedure DIPIRLJ2 (VAR P, Q: LIST; VAR reduced: BOOLEAN);
{Distributive polynomial list interreduced list in the sense of Janet. P and Q are lists of polynomials in distributive representation over an arbitrary domain, P and Q already initialised by the calling procedure. P contains polynomials which are possible candidates for prolongations, Q are already considered. Output is the list P of Janet-reduced polynomials from p plus Janet-reduced polynomials from Q. Q contains polynomials from input Q which are not Janet-reduced. reduced

is set TRUE iff a reduction took place }

procedure DIPIB4 (F: LIST): LIST;
{Distributive polynomial involutive basis. Algorithm for computing the involutive Base for a given F. Input: Distributiv Integral Polynomial List F. Output: Equivalent involutive system G.}

procedure SetDIPIBopt (options: LIST);
{Set distributive polynomial involutive base options. Input: List of 4 options in the order Trace-Level, Cancel-Option, Select-Option, ISJ-Algorithm. For details see the corresponding procedures }

procedure SetDIPIBTraceLevel (level: INTEGER);
{Set Trace Level. Input is the integer level. You get the following informatins: 0: no information, 1: computing time and number of polynomials of the computed involutive base, 2: informations about each Janet-reduction, 3: more detailed information of each Janet-reduction. }

procedure SetDIPIBCancel (Canc: CARDINAL);
{Set distributive polynomial involutive base cancel. Set the function to use to cancel down the coefficients in case of integer coefficients. Canc=1 means: use ADCAN, 2 means: use DIPMOC }

procedure SetDIPIBSelect (SEL: INTEGER);
{Set distributive polynomial involutive base select. Set polynomial selection strategy: 1: DIPSSM, 2: DIPFIRST }

procedure SetDIPIBISJ (Sel: INTEGER);
{Set distributive involutive base irreducible set in the sense of Janet. Set Janet-Irreducible-Set Algorithm, 1: DILISJ, 2: DIPIRLJ }

procedure SetDIPIBCrit (crit: INTEGER);
{Set distributive polynomial involutive base critera. Input: an integer crit. DIPIBOpt.Crit ::= TRUE iff cirt = 1 and FALSE else }

procedure SetDomainNFJ ;
{Initialize Jdomain }

15.5 DIP Integral Polynomial System in the sense of Janet.

procedure DIIPPR2 (A,B: LIST): LIST;
{Distributive integral polynomial product. A and B are distributive integral polynomials. Unlike procedure DIIPPR (in modul DIPI) B consists of one monomial. C=A*B.}

procedure DIIPNFJ (P,RPP,S: LIST): LIST;
{Distributive integral polynomial normal form in the sense of Janet. P is a list of non zero polynomials in distributive integral representation in r variables. RPP and S are distributive integral polynomials. R is a polynomial such that S is reducible to R modulo P and R is in normalform with respect to p. }

procedure DIILISJ (P: LIST): LIST;
{Distributive integral polynomial list irreducible set. P is a list of distributive integral polynomials, The result is a set such that each p element of P modulo P-(p) is in Janet-normalform }

procedure DIIPCOM (F: LIST): LIST;
{Distributive integral polynomial complete system. Subalgorithm for computing Invbase. Input: Distributive polynomial list F. Output: G: complete system, such that Ideal(G) = Ideal(F). }

procedure DIIPIB3 (F: LIST): LIST;
{Distributive integral polynomial involutive base. Algorithm for computing the involutive Base for a given F. Input: Distributiv Integral Polynomial List F. Output: Equivalent involutive system G.}

procedure DIIPIB2 (F: LIST): LIST;
{Distributive integral polynomial involutive base. Mainalgorithm for computing Invbase. Input: Distributive polynomial list F. Output: G: involutive system, such that Ideal(G) = Ideal(F). }

procedure DIIPIB (F: LIST): LIST;
{Distributive integral involutive base. Algorithm for computing the involutive Base for a given F. Input: Distributiv Integral Polynomial List F. Output: Equivalent involutive system G.}

procedure InitDIPIIB ;
{Init distributive integral involutive base. Initialization of the DIPIIB options }

procedure SetDIPIIBSelect (SEL: INTEGER);
{Set Distributive integral polynomial Select. Set polynom selection strategy }

15.6 DIP Rational Numbers Polynomial in the sense of Janet.

procedure DIRPNFJ (P,S: LIST): LIST;
{Distributive rational polynomial normal form in the sense of Janet. P is a list of non zero polynomials in distributive representation in r variables. S is a distributive polynomial. The result R is a polynomial such that S is reducible to R modulo P in the sense of Janet and R is in normalform with respect to P. }

procedure DIRLISJ (P: LIST): LIST;
{Distributive rational polynomial list irreducible set. P is a list of distributive polynomials, The result is a set of polynomials, such that each polynomial p is in Janet-normalform modulo P-(p) }

procedure DIRPCOM (F: LIST): LIST;
{Distributive rational polynom complete system. Subalgorithm for computing Invbase. Input: Distributive polynomial list F. Output: G: complete system, such that Ideal(G) = Ideal(F). }

procedure DIRPIB2 (F: LIST): LIST;
{Distributive rational polynom involutive basis. Mainalgorithm for computing Invbase. Input: Distributive polynomial list F. Output: G: involutive system, such that Ideal(G) = Ideal(F). }

procedure DIRPIB (F: LIST): LIST;
{Second Algorithm for computing the involutive Base for a given F. Input: Distributiv Rational Polynomial List F. Output: Equivalent involutive system G.}

Chapter 16

Procedure Header to Module Index

SACI	: AADV(L: LIST; VAR AL,LP: LIST);
MASSYM2	: ACOMP(A,B: LIST): LIST;
SACSYM	: ACOMP(A,B: LIST): LIST;
MASSYM2	: ACOMP1(A,B: LIST): LIST;
SACSYM	: ACOMP1(A,B: LIST): LIST;
DIPC	: AD2DIP(P: LIST): LIST;
MASADOM	: ADABSF(A:LIST):LIST;
ADTOOLS	: ADCAST(e,dd:LIST):LIST;
MASADOM	: ADCNST(A: LIST): BOOLEAN;
MASADOM	: ADCOMP(A,B:LIST):LIST;
MASADOM	: ADCONV(A,B: LIST): LIST;
CGBSYS	: ADDCGB(PLIST,P: LIST): LIST;
CGBFUNC	: ADDCON(COEFL,COND: LIST): LIST;
DIPTOOLS	: ADDDFDIL(l:LIST):LIST;
DIPTOOLS	: ADDDFDILD(l,d:LIST):LIST;
DIPTOOLS	: ADDDFDIP(p:LIST):LIST;
DIPTOOLS	: ADDDFDIPD(p,d:LIST):LIST;
ADTOOLS	: ADDDFSTR(s:ARRAY OF CHAR):LIST;
MASADOM	: ADDDREAD(): LIST;
MASADOM	: ADDDWRIT(D: LIST);
MASADOM	: ADDIF(A,B: LIST): LIST;
SYZHLP	: ADDLAST(P, PL : LIST): LIST;
DIPTOOLS	: ADDNFDIL(l:LIST):LIST;
DIPTOOLS	: ADDNFDILD(l,d:LIST):LIST;
DIPTOOLS	: ADDNFDIP(p:LIST):LIST;
DIPTOOLS	: ADDNFDIPD(p,d:LIST):LIST;
DIPAGB	: ADDNFEDIP(f: LIST): LIST;
SYZHLP	: ADDPPOS(PL, P, POS : LIST): LIST;
MASADOM	: ADEXP(A,NL: LIST): LIST;
MASADOM	: ADFACT(A: LIST): LIST;
MASADOM	: ADFACTO(A: LIST): LIST;
MASADOM	: ADFI(D,A: LIST): LIST;
MASADOM	: ADFIP(D,A: LIST): LIST;
MASADOM	: ADGCD(A,B: LIST): LIST;

SACEXT8 : AFPNIP(MB,A: LIST): LIST;
SACEXT8 : AFPPR(RL,M,A,B: LIST): LIST;
SACEXT8 : AFPQR(RL,M,A,B: LIST; VAR Q,R: LIST);
SACEXT8 : AFPRCL(M,MB,I,A: LIST): LIST;
SACEXT8 : AFPRII(M,MB,J,A,AP,DL,LP: LIST): LIST;
SACEXT8 : AFPRLS(M,MB,I,A1,A2,L1,L2: LIST; VAR LS1,LS2: LIST);
SACANF : AFPROD(M,AL,BL: LIST): LIST;
SACEXT8 : AFPRRI(M,MB,I,A,B,J,SL1,TL1: LIST): LIST;
SACEXT8 : AFPRRS(M,MB,I,A1,A2,I1,I2: LIST; VAR IS1,IS2,SL: LIST);
SACEXT8 : AFPSUM(RL,A,B: LIST): LIST;
SACANF : AFQ(M,AL,BL: LIST): LIST;
SACANF : AFSIGN(M,I,AL: LIST): LIST;
SACANF : AFSUM(AL,BL: LIST): LIST;
SACEXT8 : AFSUPB(M,A: LIST): LIST;
SACEXT8 : AFUPBA(M,A,B: LIST): LIST;
SACEXT8 : AFUPCB(M,A: LIST): LIST;
SACEXT8 : AFUPGC(M,A,B: LIST; VAR C,AB,BB: LIST);
SACEXT8 : AFUPGS(M,A: LIST): LIST;
SACEXT8 : AFUPRB(MB,I,A: LIST): LIST;
SACEXT8 : AFUPRL(M,A: LIST): LIST;
SACEXT8 : AFUPSF(M,A: LIST): LIST;
SACEXT8 : AFUPSR(M,MB,I,A,CL: LIST): LIST;
CGBFUNC : AINB(ALIST,BLIST: LIST): LIST;
SYZHLP : ALFA(L : LIST);
SYZHLP : ALFRA(L : LIST);
MASUGB : ALLELN(STAKK,L,KALT,I,PAR: LIST; VAR LF,NURLF: LIST);
MASUGB : ALLLF(STAKK,KALT,I: LIST): LIST;
SACEXT8 : ANDWR(M,I,NL: LIST);
SACEXT8 : ANFAF(M,I,AL: LIST; VAR N,J: LIST);
SACEXT8 : ANIIPE(MB,I,NB,J,TL,L: LIST; VAR S,KL,K: LIST);
SACEXT8 : ANPEDE(MB,NB: LIST; VAR TL,S,T: LIST);
SACEXT8 : ANREPE(M,MB,A,B: LIST): LIST;
MASAPF : APABS(A: LIST): LIST;
ALDPARSE : Aparse(): LIST;
MASAPF : APCMPR(A,B: LIST): LIST;
MASAPF : APCOMP(ML,EL: LIST): LIST;
MASAPF : APDIFF(A,B: LIST): LIST;
SACEXT8 : APDWR(M,I,BL,NL: LIST);
MASAPF : APEXP(A,NL: LIST): LIST;
MASAPF : APEXPT(A: LIST): LIST;
MASAPF : APFINT(N: LIST): LIST;
MASAPF : APFRN(A: LIST): LIST;
MASAPF : APLG10(A: LIST): LIST;
MASAPF : APMANT(A: LIST): LIST;
MASAPF : APNEG(A: LIST): LIST;
MASAPF : APNELD(A,B: LIST): LIST;
SYZHLP : APP0(PM : LIST): LIST;
MASAPF : APPI(): LIST;
MASAPF : APPROD(A,B: LIST): LIST;
MASAPF : APQ(A,B: LIST): LIST;
MASAPF : APROOT(A,NL: LIST): LIST;
MASAPF : APSHFT(B,EL: LIST): LIST;
MASAPF : APSIGN(A: LIST): LIST;

CGBMAIN	: CGBOUT(AC: LIST);
CGBDSTR	: CgbP(CGB: LIST): LIST;
CGBDSTR	: CgbParts(CGB: LIST; VAR P,I,VD,CD: LIST);
CGBMAIN	: CGBQFF(CGB: LIST): LIST;
CGBDSTR	: CgbVd(CGB: LIST): LIST;
CGBDSTR	: CgbWrite(CGB: LIST);
CGBSYS	: CHDEGL(PLIST: LIST): LIST;
CGBMAIN	: CHDOM(CONDS,PPS: LIST; VAR CONS,PP: LIST);
MASC	: CIM(R: LIST): LIST;
MASC	: CINT(A: LIST): LIST;
SACLIST	: CINV(L: LIST): LIST;
MASUGB	: CLF2(C,D,KLIST,NP,JP,M,L: LIST; VAR LFORM,KLISTP,J,RECLF: LIST);
MASUGB	: CLF3(C,D,KLIST,NP,JP,M: LIST; VAR LFORM,KLISTP,J: LIST);
DIPC	: CLIN(): LIST;
LISTTOOLS	: CLISTFA(atom:LIST):LIST;
MASSTOR	: CLOCK(): GAMMAINT;
clock	: ClocK(): LONGINT;
MASBIOS	: CloseBIOS();
SACLIST	: CLOUT(L: LIST);
MASBIOSU	: CLTIS(A: LIST);
CGBSYS	: CMULT(ONECL,TTERM,B: LIST): LIST;
MASC	: CNEG(R: LIST): LIST;
MASC	: CNINV(R: LIST): LIST;
MASC	: CNREAD(): LIST;
MASC	: CNWRITE(R: LIST);
CGBDSTR	: ColCons(R, W: LIST): LIST;
CGBDSTR	: ColConsCond(POL,COND: LIST): LIST;
CGBSYS	: COLDIF(T,ACOLS,COLR,COLW: LIST; VAR CRED,CWHITE: LIST);
CGBDSTR	: ColEmpty(): LIST;
CGBDSTR	: ColHT (COL: LIST): LIST;
CGBDSTR	: ColIsEmpty(Col: LIST): BOOLEAN;
CGBDSTR	: ColParts(Col: LIST; VAR R, W:LIST);
CGBDSTR	: ColpCol(COLP: LIST): LIST;
CGBDSTR	: ColpCons (POL,COL: LIST): LIST;
CGBDSTR	: ColpConsCond(POL,COND: LIST): LIST;
CGBDSTR	: ColpHT (COLP: LIST): LIST;
CGBDSTR	: ColpIsCnst(COLP: LIST): BOOLEAN;
CGBDSTR	: ColpIsZero (COLP: LIST): BOOLEAN;
CGBDSTR	: ColpParts(COLP: LIST; VAR POL,COL: LIST);
CGBDSTR	: ColpPol(COLP: LIST): LIST;
CGBSYS	: COLPRD(COL1,TTERM: LIST): LIST;
CGBDSTR	: ColRed(Col: LIST): LIST;
CGBDSTR	: ColWhite(Col: LIST): LIST;
MASSTOR	: COMP(a,L: LIST): LIST;
SACLIST	: COMP2(AL,BL,L: LIST): LIST;
SACLIST	: COMP3(AL1,AL2,AL3,L: LIST): LIST;
SACLIST	: COMP4(AL1,AL2,AL3,AL4,L: LIST): LIST;
MASUGB	: COMPA1(K,KLIST: LIST): LIST;
MASUGB	: COMPA2(K,A: LIST): LIST;
MASLISPU	: Compiledf0(F: PROCF0; VAR S: ARRAY OF CHAR);
MASLISPU	: Compiledf1(F: PROCF1; VAR S: ARRAY OF CHAR);

MASC : CSUM(R,S: LIST): LIST;
SACCOMB : CSUN(A,B: LIST): LIST;
MASBIOS : CUNIT(S : ARRAY OF CHAR): GAMMAINT;
MASUGB : CUT(TR: LIST): LIST;
SACBIOS : CWRIT2(C1,C2: GAMMAINT);
SACBIOS : CWRIT3(C1,C2,C3: GAMMAINT);
SACBIOS : CWRIT4(C1,C2,C3,C4: GAMMAINT);
SACBIOS : CWRIT5(C1,C2,C3,C4,C5: GAMMAINT);
SACBIOS : CWRIT6(C1,C2,C3,C4,C5,C6: GAMMAINT);
MASBIOS : CWRITE(C : GAMMAINT);
SACCOMB : DAND(AL,BL: LIST): LIST;
MASLISP : DCENV(E: LIST): LIST;
CGBFUNC : DCLWR(PL,B: LIST);
MASLISPU : Declare(VAR X: LIST; VAR S: ARRAY OF CHAR);
MASLISP : DECOFTAG(L: LIST): LIST;
MASLISP : DEFE(X: LIST; VAR ENV: LIST): LIST;
MASLISP : DEFF(X: LIST; VAR ENV: LIST): LIST;
MASLISP : DEFM(X: LIST; VAR ENV: LIST): LIST;
MASLISP : DEFMAP(X: LIST; VAR ENV: LIST): LIST;
MASLISP : DEFPROC(X: LIST; VAR ENV: LIST): LIST;
MASLISP : DEFRULE(X: LIST; VAR ENV: LIST): LIST;
SACD : DEGCD(AL,BL: LIST; VAR CL,UL,VL: LIST);
MASUGB : DEGRE(Q: LIST): LIST;
MASSTOR : DEQUE(L: LIST): LIST;
CGBFUNC : DET(CONDS,P: LIST; VAR DLIST,PPL: LIST);
CGBFUNC : DETPOL(GA,PI,COL: LIST; VAR DLIST,CLIST: LIST);
MASUGB : DFP(A,B: LIST): LIST;
SYZFUNC : DGBRED(GB, GBTM : LIST; VAR SY : LIST): LIST;
SACD : DGCD(AL,BL: LIST): LIST;
MASBIOS : DIBUFF();
DIPDIM : DIDIMS(G,S,U,M: LIST): LIST;
DIPDIM : DIDIMWR(DL,S,M,V: LIST);
DIPDDGB : DIDPALCMPC(AL, g1, g2, flag : LIST) : LIST;
DIPDDGB : DIDPCPLMS1(P : LIST) : LIST;
DIPDDGB : DIDPDGB(F, TF : LIST): LIST;
DIPDDGB : DIDPDNF(P,varl,g: LIST): LIST;
DIPDDGB : DIDPEGB(F, DP, TF : LIST): LIST;
DIPDDGB : DIDPELIMDGB(P : LIST) : LIST;
DIPDDGB : DIDPENF(P,varl,g: LIST): LIST;
DIPDDGB : DIDPGPOL(g1,g2: LIST): LIST;
DIPDDGB : DIDPLCPL4(P : LIST; VAR CPL, AL : LIST);
DIPDDGB : DIDPLEXTAL(AL, g : LIST) : LIST;
DIPDDGB : DIDPLM1(onestep, twostep : LIST) : LIST;
DIPDDGB : DIDPREDDGB(P : LIST) : LIST;
DIPDDGB : DIDPSPOL(g1,g2: LIST): LIST;
DIPDDGB : DIDPSPOL2(g1, g2, lcmHT, lcmHK: LIST): LIST;
DIPDDGB : DIDPTDR(P, lcmHT, pair : LIST): LIST;
DIPDDGB : DIDPUCPL1(P, g, Old : LIST) : LIST;
MASUGB : DIFF(R: LIST): LIST;
MASUGB : DIFF1(R,S: LIST): LIST;
DIPADOM : DIFIP(A,D: LIST): LIST;
CGBMISC : DIFPF(P, D: LIST; VAR DOM, VARL: LIST): LIST;
DIPRNGB : DIGBC3(B,PLI,PLJ,EL: LIST): LIST;

DIPRNGB : DIGBC4(PLI,PLJ,EL: LIST): LIST;
DIPRNGB : DIGBMI(P: LIST): LIST;
DIPROOT : DIGBSI(P,T,A: LIST): LIST;
DIPDIM : DIGBZT(S: LIST): LIST;
DIPDEC0 : DIGFET(P,IL,JL: LIST): LIST;
DIPDEC0 : DIGISM(P: LIST): LIST;
DIPDEC0 : DIGISR(P: LIST): LIST;
MASBIOS : DIGIT(C : GAMMAINT): BOOLEAN;
DIPGB : DIGMIN(P: LIST): LIST;
DIPGB : DIIFGB(P,TF: LIST): LIST;
DIPGB : DIIFLS(P: LIST): LIST;
DIPGB : DIIFMI(P: LIST): LIST;
DIPGB : DIIFNF(P,RPP,S: LIST): LIST;
DIPI : DIIFRP(A: LIST): LIST;
DIPGB : DIIFSP(A,B: LIST): LIST;
DIPIGB : DIIGBA(PL,P,TF: LIST): LIST;
DIPIGB : DIIGMI(P: LIST): LIST;
DIPI : DIILFR(A: LIST): LIST;
DIPI : DIILFRCD(A: LIST): LIST;
CGBMISC : DIILIS(P: LIST): LIST;
DIPIGB : DIILIS(P: LIST): LIST;
DIPI : DIILRD(V: LIST): LIST;
DIPI : DIILWR(A,V: LIST);
DIPI : DIIPAB(A: LIST): LIST;
DIPIDGB : DIIPALCMPC(AL, g1, g2, flag : LIST) : LIST;
DIPI : DIIPCP(A: LIST; VAR CL,AP: LIST);
DIPIDGB : DIIPCPLMS1(P : LIST) : LIST;
DIPI : DIIPDF(A,B: LIST): LIST;
DIPIDGB : DIIPDGB(F, TF : LIST): LIST;
DIPI : DIIPDM(A: LIST): LIST;
DIPIDGB : DIIPDNF(P,varl,g: LIST): LIST;
DIPI : DIIPDR(A,IL: LIST): LIST;
DIPIDGB : DIIPEGB(F, TF : LIST): LIST;
DIPIDGB : DIIPELIMDGB(P : LIST) : LIST;
DIPI : DIIPEM(A,AL: LIST): LIST;
DIPIDGB : DIIPENF(P,varl,g: LIST): LIST;
DIPI : DIIPEV(A,IL,AL: LIST): LIST;
DIPI : DIIPEX(A,NL: LIST): LIST;
DIPIGB : DIIPGB(P,TF: LIST): LIST;
DIPIDGB : DIIPGPOL(g1,g2: LIST): LIST;
DIPI : DIIPHD(A,IL,NL: LIST): LIST;
DIPI : DIIPIP(A,BL: LIST): LIST;
DIPI : DIIPIQ(A,BL: LIST): LIST;
DIPIDGB : DIIPLCPL4(P : LIST; VAR CPL, AL : LIST);
DIPIDGB : DIIPLEXTAL(AL, g : LIST) : LIST;
DIPIDGB : DIIPLM1(onestep, twostep : LIST) : LIST;
DIPI : DIIPLS(A: LIST): LIST;
DIPI : DIIPMN(A: LIST): LIST;
DIPIGB : DIIPNF(P,RPP,S: LIST): LIST;
DIPI : DIIPNG(A: LIST): LIST;
CGBMISC : DIIPNORM(P: LIST): LIST;
DIPI : DIIPON(A: LIST): LIST;
DIPI : DIIPPR(A,B: LIST): LIST;

```
DIPI          : DIIPPS(A,B: LIST): LIST;
DIPI          : DIIPQ(A,B: LIST): LIST;
DIPI          : DIIPQR(A,B: LIST; VAR Q,R: LIST);
DIPI          : DIIPRA(RL,KL,LL,EL: LIST): LIST;
DIPI          : DIIPRD(V: LIST): LIST;
DIPIDGB       : DIIPREDDGB(P : LIST) : LIST;
DIPI          : DIIPSG(A: LIST): LIST;
DIPI          : DIIPSM(A,B: LIST): LIST;
DIPI          : DIIPSN(A: LIST): LIST;
DIPI          : DIIPSO(A: LIST): LIST;
DIPIGB        : DIIPSP(A,B: LIST): LIST;
DIPIDGB       : DIIPSPOL(g1,g2: LIST): LIST;
DIPIDGB       : DIIPSPOL2(g1, g2, lcmHT, lcmHK: LIST): LIST;
DIPI          : DIIPSU(A,IL,B: LIST): LIST;
DIPI          : DIIPSV(A,B: LIST): LIST;
DIPIDGB       : DIIPTDR(P, lcmHT, pair : LIST): LIST;
DIPI          : DIIPTM(A,HL: LIST): LIST;
DIPI          : DIIPTR(A,HL,IL: LIST): LIST;
DIPIDGB       : DIIPUCPL1(P, g, Old : LIST) : LIST;
DIPI          : DIIPWR(A,V: LIST);
DIPI          : DIIPWV(A: LIST);
DIPI          : DIIRAS(RL,KL,LL,EL,QL: LIST): LIST;
DIPROOT       : DIITNT(T: LIST): LIST;
DIPROOT       : DIITWR(TP,EPS: LIST);
DIPC          : DILBSO(A: LIST);
DIPTOOLS      : DILCONV(P,E: LIST):LIST;
DIPRNGB       : DILCPL(P: LIST; VAR D,B: LIST);
DIPDIM        : DILDIM(G: LIST; VAR DL,S,M: LIST);
DIPTOOLS      : DILFDILP(L,NewDd:LIST):LIST;
MASNCC        : DILFEL(a, E: LIST): LIST;
CGBMISC       : DILFPFL(PFL, D: LIST; VAR DOM, VARL: LIST): LIST;
DIPC          : DILFPL(RL,A: LIST): LIST;
DIPTOOLS      : DILIMO(P:LIST):LIST;
DIPTOOLS      : DILINV(dil,j,k:LIST):LIST;
DIPGB         : DILIS(P: LIST): LIST;
DIPTOOLS      : DILMOC(L:LIST):LIST;
DIPC          : DILPERM(dil,perm: LIST):LIST;
DIPTOOLS      : DILPFDIL(L,r,newdd:LIST):LIST;
DIPTOOLS      : DILPROD(L:LIST;domain:LIST):LIST;
DIPADOM       : DILRD(V,D: LIST): LIST;
DIPADOM       : DILSUM(A: LIST): LIST;
DIPRNGB       : DILUPL(PL,P,D,B: LIST): LIST;
DIPADOM       : DILWR(A,V: LIST);
CGBAPPL       : DIMEXE(GS,V: LIST): LIST;
CGBMAIN       : DIMIS(PL,VL: LIST; VAR MAXVL: LIST): LIST;
DIPTOOLS      : DIMPAD(c,ev,i:LIST):LIST;
MASNCGB       : DIN1GB(T,P,TF: LIST): LIST;
MASNCC        : DINCCO(T, A, B: LIST): LIST;
MASNCC        : DINCCP(T, E: LIST): LIST;
MASNCC        : DINCCPpre(T, E: LIST): LIST;
MASNCGB       : DINCGB(T,P,TF: LIST): LIST;
MASNCGB       : DINLGB(T,P,TF: LIST): LIST;
MASNCGB       : DINLGM(T,P: LIST): LIST;
```

```
DIPC            : DIPINV(A,JL,KL: LIST): LIST;
DIPADOM         : DIPIRL(VAR P: LIST; VAR CS: BOOLEAN);
DIPC            : DIPLBC(A: LIST): LIST;
DIPC            : DIPLDC(A: LIST): LIST;
DIPTOO          : DIPLDM(A: LIST): LIST;
DIPIDEAL        : DIPLDV(A,V: LIST): LIST;
CGBMISC         : DIPLFPFL (PFL: LIST; VAR DOM, VARL: LIST): LIST;
DIPADOM         : DIPLIR(P: LIST): LIST;
DIPC            : DIPLM(L1,L2: LIST): LIST;
DINNGB          : DIPLMD(P:LIST):LIST;
DIPC            : DIPLPM(A: LIST): LIST;
DIPC            : DIPLRS(A: LIST);
DIPC            : DIPMAD(A: LIST; VAR AL,EL,AP: LIST);
MASUGB          : DIPMC2(A,C,P: LIST): LIST;
DIPC            : DIPMCP(AL,EL,A: LIST): LIST;
DIPADOM         : DIPMOC(A: LIST): LIST;
DIPC            : DIPMPM(A,PL: LIST): LIST;
DIPC            : DIPMPV(A,SL,PL: LIST): LIST;
DIPC            : DIPMRD(A: LIST): LIST;
DIPC            : DIPMST(A,AL,EL: LIST);
DIPTOOLS        : DIPMVV(p: LIST):LIST;
DIPC            : DIPNBC(A: LIST): LIST;
DIPADOM         : DIPNEG(A: LIST): LIST;
DIPADOM         : DIPNF(A,B: LIST): LIST;
DIPGB           : DIPNOR(P,S: LIST): LIST;
DIPC            : DIPNOV(A: LIST): LIST;
DIPTOOLS        : DIPONE(d:LIST):LIST;
DIPTOOLS        : DIPPAD(p,i: LIST):LIST;
DIPTOOLS        : DIPPCPP(P:LIST; VAR content,ppt: LIST);
DIPTOOLS        : DIPPFDIP(p,r,NewDd:LIST;VAR q,vlist: LIST);
DIPTOOLS        : DIPPOWER(p,n:LIST):LIST;
DIPADOM         : DIPQR(A,B: LIST; VAR Q,R: LIST);
DIPC            : DIPRED(A: LIST): LIST;
DIPADOM         : DIPRLF(P,p: LIST): LIST;
DIPADOM         : DIPROD(A,B: LIST): LIST;
DIPAGB          : DIPRWTDG(f,W: LIST): LIST;
DIPADOM         : DIPS(A,B: LIST): LIST;
DIPADOM         : DIPSFF(A,VOO: LIST): LIST;
DIPGB           : DIPSP(A,B: LIST): LIST;
DINNGB          : DIPSPS(D,B:LIST):LIST;
DIPADOM         : DIPSUM(A,B: LIST): LIST;
DIPC            : DIPTBC(A: LIST): LIST;
DIPC            : DIPTCF(A: LIST): LIST;
DIPC            : DIPTCS(A,IL: LIST): LIST;
DIPC            : DIPTDG(A: LIST): LIST;
MASYMDIP        : DIPTODEF(T: LIST): LIST;
DIPTOO          : DIPTRM(A: LIST): LIST;
DIPTOO          : DIPTYP(A: LIST): LIST;
DIPC            : DIPUNT(A: LIST): LIST;
DIPC            : DIPUV(A: LIST): LIST;
MASYMDIP        : DIPVDEF(V: LIST): LIST;
DIPTOO          : DIPVOP(P,V: LIST; VAR PP,VP: LIST);
DIPTOO          : DIPVOPP(P,V: LIST; VAR PP,VP,PV: LIST);
```

DIPTOOLS : DIPXCM(p,mvars: LIST):LIST;
DIPADOM : DIREAD(V,D: LIST): LIST;
TIPRNGB : DIREGB(P, TF: LIST; VAR GB, GBM: LIST);
DIPGCD : DIRFAC(P: LIST): LIST;
DIPRN : DIRFIP(A: LIST): LIST;
DIPRNGB : DIRGBA(PL,P,TF: LIST): LIST;
DIPRNGB : DIRGBR(P,TF: LIST): LIST;
DIPDEC0 : DIRGZS(VB,PB,W: LIST): LIST;
DIPIDEAL : DIRLCT(A,B: LIST): LIST;
DIPIDEAL : DIRLIP(PL,A,B: LIST): LIST;
DIPRNGB : DIRLIS(P: LIST): LIST;
DIPDEC0 : DIRLPD(A,VP: LIST): LIST;
DIPIDEAL : DIRLPI(A,P,VP: LIST): LIST;
DIPDEC0 : DIRLPW(A,V,L: LIST);
DIPRN : DIRLRD(V: LIST): LIST;
DIPRN : DIRLWR(A,V,S: LIST);
DIPZ : DIRMPG(IL,F: LIST): LIST;
DIPROOT : DIROWR(V,P,EPS: LIST);
DIPRN : DIRPAB(A: LIST): LIST;
DIPDEC0 : DIRPDA(A,VP: LIST): LIST;
DIPRN : DIRPDF(A,B: LIST): LIST;
DIPRN : DIRPDM(A: LIST): LIST;
DIPRN : DIRPDR(A,IL: LIST): LIST;
DIPRN : DIRPEM(A,AL: LIST): LIST;
SYMMFU : DIRPES(RL: LIST): LIST;
DIPRN : DIRPEV(A,IL,AL: LIST): LIST;
DIPRN : DIRPEX(A,NL: LIST): LIST;
MASYMDIP : DIRPFT(T, V: LIST): LIST;
DIPRNGB : DIRPGB(P,TF: LIST): LIST;
DIPRN : DIRPHD(A,IL,NL: LIST): LIST;
DIPRN : DIRPLS(A: LIST): LIST;
DIPRN : DIRPMC(A: LIST): LIST;
DIPRN : DIRPMN(A: LIST): LIST;
DIPRNGB : DIRPNF(P,S: LIST): LIST;
DIPRN : DIRPNG(A: LIST): LIST;
DIPRN : DIRPON(A: LIST): LIST;
DIPRN : DIRPPR(A,B: LIST): LIST;
DIPRN : DIRPQ(A,B: LIST): LIST;
DIPRN : DIRPQR(A,B: LIST; VAR Q,R: LIST);
DIPRN : DIRPRA(RL,KL,LL,EL: LIST): LIST;
DIPRN : DIRPRD(V: LIST): LIST;
DIPRN : DIRPRP(A,BL: LIST): LIST;
DIPRN : DIRPRQ(A,BL: LIST): LIST;
SYMMFU : DIRPSE(Q,U: LIST; VAR PL,V: LIST);
DIPRN : DIRPSG(A: LIST): LIST;
DIPRN : DIRPSM(A,B: LIST): LIST;
DIPRN : DIRPSN(A: LIST): LIST;
DIPRN : DIRPSO(A: LIST): LIST;
MASUGB : DIRPSP(A,B,X: LIST): LIST;
DIPRNGB : DIRPSP(A,B: LIST): LIST;
SYMMFU : DIRPSR(Q,PL: LIST; VAR P1,P2: LIST);
DIPRN : DIRPSU(A,IL,B: LIST): LIST;
DIPRN : DIRPSV(A,B: LIST): LIST;

```
DIPRN           : DIRPTM(A,HL: LIST): LIST;
DIPRN           : DIRPTR(A,HL,IL: LIST): LIST;
DIPRN           : DIRPWR(A,V,S: LIST);
DIPRN           : DIRPWV(A: LIST);
DIPRN           : DIRRAS(RL,KL,LL,EL,QL: LIST): LIST;
MASUGB          : DIRRNF(P,S,X,V: LIST): LIST;
DIPDEC0         : DITFZS(N: LIST): LIST;
DIPDEC0         : DITSPL(T: LIST; VAR T0,T1: LIST);
DIPADOM         : DIWRIT(A,V: LIST);
SACD            : DLOG2(AL: LIST): LIST;
DIPTOO          : DMEVAD(A,E: LIST): LIST;
MASLISP         : DMIA(X: LIST; VAR ENV: LIST): LIST;
SACDPOL         : DMPPRD(RL,ML,A,B: LIST): LIST;
SACDPOL         : DMPSUM(RL,ML,A,B: LIST): LIST;
SACDPOL         : DMUPNR(PL,A,B: LIST): LIST;
SACCOMB         : DNIMP(AL,BL: LIST): LIST;
DINNGB          : DNLCPL(EL,P: LIST; VAR D,B: LIST);
DINNGB          : DNLUPL(EL,PL,P,D,B: LIST): LIST;
DINNGB          : DNN2GB(EL,P,TF: LIST): LIST;
DINNGB          : DNNLES(EL,P:LIST):LIST;
DINNGB          : DNNLGB(EL,P,TF: LIST): LIST;
DINNGB          : DNNLIS(EL,P: LIST): LIST;
DINNGB          : DNNLNF(EL,P,S:LIST):LIST;
DINNGB          : DNNLSP(EL,A,B:LIST):LIST;
DINNGB          : DNNRES(EL,P,DL,DP:LIST):LIST;
DINNGB          : DNNRGB(EL,P,TF: LIST): LIST;
DINNGB          : DNNRIS(EL,P: LIST): LIST;
DINNGB          : DNNRNF(EL,P,S:LIST):LIST;
DINNGB          : DNNRSP(EL,A,B:LIST):LIST;
DINNGB          : DNNTGB(EL,P,TF:LIST):LIST;
SACCOMB         : DNOT(AL: LIST): LIST;
DINNGB          : DNRCPL(EL,P: LIST; VAR D,B: LIST);
DINNGB          : DNRUPL(EL,PL,P,D,B: LIST): LIST;
MASUGB          : DO1(LFP: LIST): LIST;
DOMAF           : DomLoadAF();
DOMAPF          : DomLoadAPF();
DOMC            : DomLoadC();
DOMFF           : DomLoadFF();
DOMI            : DomLoadI();
DOMIP           : DomLoadIP();
DOMMD           : DomLoadMD();
DOMMI           : DomLoadMI();
DOMO            : DomLoadO();
DOMQ            : DomLoadQ();
DOMRF           : DomLoadRF();
DOMRN           : DomLoadRN();
DOMRP           : DomLoadRP();
MASADOM         : DomSummary();
MASU            : DoParse(): LIST;
SACCOMB         : DOR(AL,BL: LIST): LIST;
MASBIOSU        : DOS(A: LIST): LIST;
MASmtc          : DOS(s: ARRAY OF CHAR): INTEGER;
MASU            : DoWrite(Y: LIST);
```

MASSYM2	: ENTER(L: LIST): LIST;
SACSYM	: ENTER(L: LIST): LIST;
DIPC	: EPREAD(): LIST;
CGBFUNC	: EQPLCL(ALIST,BLIST: LIST): LIST;
SACLIST	: EQUAL(AL,BL: LIST): LIST;
MASERR	: ERROR(a: GAMMAINT; s: ARRAY OF CHAR);
MASERR	: ErrorHandler(a: P0): GAMMAINT;
MASBIOS	: EStreamKind(): INTEGER;
MASSPEC	: EVALUATE(X: LIST; VAR ENV: LIST): LIST;
SYMMFU	: EVASC(U: LIST): LIST;
DIPC	: EVCADD(U,IL,EL: LIST; VAR V,FL: LIST);
DIPTOOLS	: EVCNSTR(ev,mvars:LIST):BOOLEAN;
DIPC	: EVCOMP(U,V: LIST): LIST;
MASUGB	: EVCOMP(U,V: LIST): LIST;
DIPC	: EVCSUB(U,IL,EL: LIST; VAR V,FL: LIST);
DIPC	: EVDEL(U,IL: LIST; VAR V,EL: LIST);
DIPC	: EVDER(U,IL,EL: LIST; VAR V,FL: LIST);
DIPC	: EVDFSI(U,V: LIST; VAR W,SL: LIST);
DIPC	: EVDIF(U,V: LIST): LIST;
DIPC	: EVDOV(U: LIST): LIST;
DIPC	: EVEXC(U,IL,JL: LIST): LIST;
DIPTOOLS	: EVEXT(p,evx:LIST):LIST;
SYZHLP	: EVF(EV, L : LIST): LIST;
DIPDIM	: EVGBIT(S,G: LIST): LIST;
MASNCC	: EVGCD(U,V: LIST): LIST;
DIPC	: EVIGLC(U,V: LIST): LIST;
DIPC	: EVILCI(U,V: LIST): LIST;
DIPC	: EVILCP(U,V: LIST): LIST;
MASNCC	: EVINV(U,i,k: LIST): LIST;
DIPC	: EVITDC(U,V: LIST): LIST;
SYZHLP	: EVL(PM : LIST) : LIST;
DIPC	: EVLCM(U,V: LIST): LIST;
DIPC	: EVLFCP(L,U,V: LIST): LIST;
MASUGB	: EVLFCP(L,U,V: LIST): LIST;
MASNCC	: EVLGIL(D: LIST): LIST;
MASNCC	: EVLGTD(r,d,L: LIST): LIST;
MASNCC	: EVLINV(L,i,k: LIST): LIST;
DINNGB	: EVLLCM(EL,S,T:LIST):LIST;
DINNGB	: EVLRCM(EL,S,T:LIST):LIST;
MASUGB	: EVLRNBSO(A: LIST);
DIPC	: EVMT(U,V: LIST): LIST;
DINNGB	: EVNCLD(EL,S,T:LIST):LIST;
DINNGB	: EVNCRD(EL,S,T:LIST):LIST;
DINNGB	: EVNLDT(EL,S,T:LIST):LIST;
DINNGB	: EVNNCP(EL,S,T: LIST): LIST;
DIPC	: EVNNZE(U: LIST): LIST;
DINNGB	: EVNRDT(EL,S,T:LIST):LIST;
DIPTOOLS	: EvordPop();
DIPTOOLS	: EvordPush(evord: LIST);
CGBMISC	: EvordReset();
CGBMISC	: EvordSet(T: LIST);
DIPC	: EvordWrite();
DIPC	: EVOWRITE(EVO: LIST);

MASFF : FFSUM(p,M,AL,BL: LIST): LIST;
MASFF : FFWRITE(R, V: LIST);
CGBMISC : FILWRITE(L: LIST; N:INTEGER);
CGBSYS : FINCOL(APP,ACOLS,COLR,COLW: LIST; VAR CRED,CWHITE: LIST);
CGBFUNC : FINDBC(RE,POL: LIST): LIST;
CGBFUNC : FINDCP(TTERM,WHITE: LIST): LIST;
CGBSYS : FINDPI(PCO,P: LIST; VAR PCI,RE: LIST);
CGBSYS : FINDPITOP(PCO,P: LIST; VAR PCI,RE: LIST);
CGBFUNC : FINDRM(RE,POL: LIST): LIST;
MASSTOR : FIRST(L: LIST): LIST;
SACLIST : FIRST2(L: LIST; VAR AL,BL: LIST);
SACLIST : FIRST3(L: LIST; VAR AL1,AL2,AL3: LIST);
SACLIST : FIRST4(L: LIST; VAR AL1,AL2,AL3,AL4: LIST);
MASF : FLOG10(F: MFLOAT): MFLOAT;
CGBMISC : FLWRITE(L: LIST);
MASREP : ForEachinList(r, f, E: LIST): LIST;
MASREP : ForEachinRep(r, f, E: LIST): LIST;
CGBDSTR : FormFCond(Cond: LIST; VAR VD: LIST): LIST;
SACLIST : FOURTH(L: LIST): LIST;
SACPRIM : FRESL(NL: LIST; VAR ML,L: LIST);
SACPRIM : FRLSM(ML,AL: LIST): LIST;
MASREP : FullRep(r: LIST): LIST;
CGBSYS : GBDIFF(COND,A,ACOLS,B,BCOLS: LIST; VAR C,CCOLS: LIST);
SYZGB : GBE(PL, SANZ, L : LIST): LIST;
SYZGB : GBEF(PL, SANZ, L : LIST; VAR GBTM : LIST): LIST;
SYZGB : GBF(PL, SANZ: LIST; VAR GBTM : LIST): LIST;
CGBAPPL : GBHELP(COND,PPAIRS,P: LIST; VAR C0,C1: LIST);
CGBSYS : GBSYS(CNDS,P: LIST): LIST;
CGBSYS : GBSYSF(CNDS,P: LIST): LIST;
SYZHLP : GBTMRED(GBTM, POSV : LIST) : LIST;
CGBSYS : GBUPD(COND,P,GBSYS: LIST): LIST;
DIPROOT : GBZSET(V,PP,EPS: LIST);
SACPRIM : GDPGEN(ML: LIST; KL: INTEGER): LIST;
MASSYM : GENARRAY(A: LIST): LIST;
MASSYM : GENINDEX(A: LIST): LIST;
MASLISP : GENPL(P,V,T,D: LIST): LIST;
SYZHLP : GENPOSV(GB, GBR : LIST): LIST;
MASSYM2 : GENSYM(): LIST;
SACSYM : GENSYM(): LIST;
MASLISP : GENTE(Z,N,D: LIST): LIST;
MASSYM2 : GET(S,AL: LIST): LIST;
SACSYM : GET(S,AL: LIST): LIST;
MASREP : GetRep(n,r: LIST): LIST;
MASmtc : getstck(): ADDRESS;
MASmtc : gettoc(): ADDRESS;
CGBMAIN : GGREEN(GS: LIST);
GSYMFUIN : GINBAS(PG: LIST): LIST;
GSYMFUIN : GINCHK(PG, BASE, POL: LIST): LIST;
GSYMFUIN : GINCHKBAS(VAR BASE, POL: LIST);
GSYMFUIN : GINCUT(PG, POL: LIST; VAR POL_1, POL_2: LIST);
GSYMFUIN : GINOPL(PG, ML: LIST): LIST;
GSYMFUIN : GINORP(PG, MO: LIST): LIST;

SACCOMB : IBCOEF(NL,KL: LIST): LIST;
SACCOMB : IBCPS(NL,KL: LIST): LIST;
SACI : ICOMP(A,B: LIST): LIST;
SACI : IDEGCD(AL,BL: LIST; VAR CL,UL1,VL1,UL2,VL2: LIST);
SACI : IDIF(A,B: LIST): LIST;
SACI : IDIPR2(A,B,AL,BL: LIST): LIST;
SACI : IDP2(A,KL: LIST): LIST;
SACI : IDPR(A,BL: LIST): LIST;
SACI : IDQ(A,BL: LIST): LIST;
SACI : IDQR(A,BL: LIST; VAR Q,RL: LIST);
SACI : IDREM(A,BL: LIST): LIST;
SACI : IEGCD(AL,BL: LIST; VAR CL,UL1,VL1: LIST);
SYZMAIN : IEQ(PL, P, SANZ, SRD : LIST): LIST;
SACI : IEVEN(A: LIST): BOOLEAN;
SACI : IEXP(A,NL: LIST): LIST;
SACPRIM : IFACT(NL: LIST): LIST;
SACCOMB : IFACTL(NL: LIST): LIST;
SACI : IFCL2(AL: LIST; VAR ML,NL: LIST);
MASF : IFF(F: MFLOAT): LIST;
DIPRF : IFWRIT(R,V: LIST);
SACI : IGCD(A,B: LIST): LIST;
SACI : IGCDCF(A,B: LIST; VAR C,AB,BB: LIST);
SACI : IHEGCD(A,B: LIST; VAR C,V: LIST);
SACROOT : IIC(A,AP,I,L: LIST): LIST;
DIPE : IJACS(X,Y: LIST): LIST;
LINALGI : IKM(A, B : LIST): LIST;
DIPE : ILADDC(U,CL: LIST): LIST;
SACI : ILCM(A,B: LIST): LIST;
SACI : ILCOMB(A,B,UL,VL: LIST): LIST;
DIPE : ILEXPR(U,V: LIST; VAR W,SL: LIST);
DIPE : ILILPR(U,V: LIST; VAR W,SL: LIST);
DIPE : ILINPR(U,V: LIST; VAR W,SL: LIST);
DIPE : ILIRPR(U,V: LIST; VAR W,SL: LIST);
MASAPF : ILOG10(N: LIST): LIST;
SACI : ILOG2(A: LIST): LIST;
SACPRIM : ILPDS(NL,AL,BL: LIST; VAR PL,NLP: LIST);
DIPE : ILSCMP(U,V: LIST): LIST;
DIPE : ILWCMP(U,V: LIST): LIST;
SACI : ILWRIT(L: LIST);
SACI : IMAX(AL,BL: LIST): LIST;
LINALGI : IMDET(M : LIST): LIST;
LINALGI : IMDETL(M : LIST): LIST;
LINALGI : IMDIF(A, B : LIST): LIST;
LINALGI : IMFRNM(A : LIST): LIST;
LINALGI : IMFRNM1(A, B : LIST; VAR C, D: LIST);
LINALGI : IMGE(M : LIST): LIST;
LINALGI : IMGELUD(M : LIST; VAR L, U: LIST);
SACI : IMIN(AL,BL: LIST): LIST;
LINALGI : IMLT(L, b : LIST): LIST;
LINALGI : IMMAX(M : LIST): LIST;
SACI : IMP2(A,HL: LIST): LIST;
SACPRIM : IMPDS(NL,AL,BL: LIST; VAR PL,QL: LIST);
LINALGI : IMPROD(A, B : LIST): LIST;

SACPFAC : IPFAC(RL,A: LIST; VAR SL,CL,L: LIST);
SACIPOL : IPFCB(V: LIST): LIST;
SACUPFAC : IPFLC(RL,M,I,A,L,D: LIST): LIST;
MASPGCD : IPFLMER(r,F1,F2: LIST): LIST;
SACIPOL : IPFRP(RL,A: LIST): LIST;
SACROOT : IPFSD(RL,A: LIST): LIST;
SACEXT6 : IPFSFB(RL,A: LIST): LIST;
SACPGCD : IPGCDC(RL,A,B: LIST; VAR C,AB,BB: LIST);
SACPFAC : IPGFCB(RL,A: LIST): LIST;
SACIPOL : IPGSUB(RL,A,SL,L: LIST): LIST;
SACIPOL : IPHDMV(RL,A,KL: LIST): LIST;
SACPGCD : IPIC(RL,A: LIST): LIST;
SACPGCD : IPICPP(RL,A: LIST; VAR AL,AB: LIST);
SACPGCD : IPICS(RL,A,CL: LIST): LIST;
SACIPOL : IPIHOM(RL,D,A: LIST): LIST;
SACEXT4 : IPINT(RL,A,BL: LIST): LIST;
SACIPOL : IPIP(RL,AL,B: LIST): LIST;
SACPGCD : IPIPP(RL,A: LIST): LIST;
SACIPOL : IPIPR(RL,D,A,B: LIST): LIST;
SACIPOL : IPIQ(RL,A,BL: LIST): LIST;
SACPFAC : IPIQH(RL,PL,D,AB,BB,SB,TB,M,C: LIST; VAR A,B: LIST);
DIPGCD : IPLCM(RL,A,B: LIST): LIST;
SACEXT5 : IPLCPP(RL,A: LIST; VAR C,P: LIST);
SACROOT : IPLRRI(L: LIST): LIST;
SACIPOL : IPMAXN(RL,A: LIST): LIST;
SACIPOL : IPNEG(RL,A: LIST): LIST;
SACIPOL : IPONE(RL,A: LIST): LIST;
SACI : IPOWER(A,L: LIST; VAR B,NL: LIST);
DINNGB : IPOWER(EL,AL:LIST):LIST;
SACPGCD : IPPGSD(RL,A: LIST): LIST;
SACPGCD : IPPP(RL,A: LIST): LIST;
SACIPOL : IPPROD(RL,A,B: LIST): LIST;
SACEXT5 : IPPSC(RL,A,B: LIST): LIST;
SACIPOL : IPPSR(RL,A,B: LIST): LIST;
SACIPOL : IPQ(RL,A,B: LIST): LIST;
SACIPOL : IPQR(RL,A,B: LIST; VAR Q,R: LIST);
DIPIPOL : IPRAN(RL,KL,QL,N: LIST): LIST;
SACIPOL : IPRAN(RL,KL,QL,N: LIST): LIST;
SACROOT : IPRCH(A,I,KL: LIST): LIST;
SACROOT : IPRCHS(A,I,KL: LIST): LIST;
SACROOT : IPRCN1(A,I,SL,KL: LIST): LIST;
SACROOT : IPRCNP(A,I: LIST; VAR SLP,SLPP,J: LIST);
SACIPOL : IPREAD(VAR RL,A,V: LIST);
SACPGCD : IPRES(RL,A,B: LIST): LIST;
SACEXT7 : IPRICL(A: LIST): LIST;
SACROOT : IPRIM(A: LIST): LIST;
SACROOT : IPRIMO(A,AP,I: LIST): LIST;
SACROOT : IPRIMS(A,AP,I: LIST): LIST;
SACROOT : IPRIMU(A: LIST): LIST;
SACROOT : IPRIU(A: LIST): LIST;
SACROOT : IPRIUP(A: LIST): LIST;
MASI : IPROD(A,B: LIST): LIST;
SACI : IPROD(A,B: LIST): LIST;

SACIPOL : IUPBES(A,AL: LIST): LIST;
SACIPOL : IUPBHT(A,KL: LIST): LIST;
SACIPOL : IUPBRE(A,AL: LIST): LIST;
SACIPOL : IUPCHT(A: LIST): LIST;
SACUPFAC : IUPFAC(A: LIST; VAR SL,CL,L: LIST);
SACUPFAC : IUPFDS(A: LIST; VAR PL,F,C: LIST);
SACEXT4 : IUPIHT(A,NL: LIST): LIST;
SACEXT8 : IUPMRN(R: LIST): LIST;
SACIPOL : IUPNT(A: LIST): LIST;
SACUPFAC : IUPQH(PL,AB,BB,SB,TB,M,C: LIST; VAR A,B: LIST);
SACUPFAC : IUPQHL(PL,F,M,C: LIST): LIST;
SACROOT : IUPRB(A: LIST): LIST;
SACEXT5 : IUPRC(A,B: LIST; VAR C,R: LIST);
SACROOT : IUPRLP(A: LIST): LIST;
SACIPOL : IUPTPR(NL,A,B: LIST): LIST;
SACIPOL : IUPTR(A,HL: LIST): LIST;
SACIPOL : IUPTR1(A: LIST): LIST;
SACROOT : IUPVAR(A: LIST): LIST;
SACEXT7 : IUPVOI(A,I: LIST): LIST;
SACROOT : IUPVSI(A,I: LIST): LIST;
SACUPFAC : IUSFPF(A: LIST): LIST;
LINALGI : IVFRNV(A: LIST): LIST;
LINALGI : IVFRNV1(A, B : LIST; VAR C, D: LIST);
DIPE : IVHOM(U,IL,JL: LIST): LIST;
LINALGI : IVLC(a, A, b, B : LIST): LIST;
LINALGI : IVMAX(M : LIST): LIST;
DIPE : IVRAND(KL,QL,NL: LIST): LIST;
LINALGI : IVSPROD(A, B : LIST): LIST;
LINALGI : IVSQ(a, A: LIST): LIST;
LINALGI : IVSSUM(A : LIST): LIST;
LINALGI : IVSVPROD(A, B : LIST): LIST;
LINALGI : IVSVSUM(A, B : LIST): LIST;
LINALGI : IVVDIF(A, B : LIST): LIST;
LINALGI : IVVPROD(A, B : LIST): LIST;
LINALGI : IVVSUM(A, B : LIST): LIST;
LINALGI : IVWRITE(A : LIST);
SACI : IWRITE(A: LIST);
DIPDIM : IXSUBS(V,I: LIST): LIST;
CGBSYS : KEYCOL(EL,ACOLS: LIST; VAR KEY,ALIST: LIST);
DIPE : KREISP(NL: LIST): LIST;
MASLISP : LAMBDAP(S: LIST): BOOLEAN;
SACLIST : LAST(L: LIST): LIST;
MASUGB : LASTEL(Y: LIST): LIST;
SACSET : LBIBMS(L: LIST): LIST;
SACSET : LBIBS(L: LIST);
SACSET : LBIM(L1,L2: LIST): LIST;
DIPTOO : LBLXCO(U,V: LIST): LIST;
SACEXT1 : LCONC(L: LIST): LIST;
MASUGB : LDEG(L: LIST): LIST;
DIPAGB : LDIPEXTEND(P: LIST): LIST;
SACLDIO : LDSMKB(A,BL: LIST; VAR XLS,N: LIST);
SACLDIO : LDSSBR(A,BL: LIST; VAR XLS,N: LIST);
DIPAGB : LECPUNEXTEND(P: LIST): LIST;

MASBIOS	: MASORDI(C : GAMMAINT): GAMMAINT;
MASELEM	: MASQREM(a,b: GAMMAINT; VAR q,r: GAMMAINT);
masreadline	: masReadL(VAR s: tString);
maskpathsea	: masReadOpen(file: ARRAY OF CHAR): IO_tFile;
MASELEM	: MASREM(a,b: GAMMAINT): GAMMAINT;
MASELEM	: MASSIGN(a: GAMMAINT): GAMMAINT;
CGBAPPL	: MCOEF(PCO: LIST; VAR COEFLI,COEF,TL: LIST);
SACMUFAC	: MCPMV(NL,L: LIST): LIST;
SACM	: MDCRA(ML1,ML2,MLP1,AL1,AL2: LIST): LIST;
SACM	: MDDIF(ML,AL,BL: LIST): LIST;
LINALGRN	: MDELCOL(M, i : LIST): LIST;
SACM	: MDEXP(ML,AL,NL: LIST): LIST;
SACM	: MDHOM(ML,A: LIST): LIST;
LINALGRN	: MDIM(M : LIST): LIST;
SACM	: MDINV(ML,AL: LIST): LIST;
SACM	: MDLCRA(ML1,ML2,L1,L2: LIST): LIST;
SACM	: MDNEG(ML,AL: LIST): LIST;
SACM	: MDPROD(ML,AL,BL: LIST): LIST;
SACM	: MDQ(ML,AL,BL: LIST): LIST;
SACM	: MDRAN(ML: LIST): LIST;
SACM	: MDSUM(ML,AL,BL: LIST): LIST;
DIPE	: MDVHOM(ML,U: LIST): LIST;
SACLIST	: MEMBER(AL,L: LIST): LIST;
MASSYM	: MEMQ(AL,L: LIST): BOOLEAN;
NOETHER	: MERGE(FLAG: BOOLEAN; BASE1, POL1: LIST; VAR BASE2, POL2: LIST);
MASUGB	: MERGE(STALT,STNEU: LIST): LIST;
LINALGRN	: MFILL(M, m, n: LIST): LIST;
SYZGB	: MGB(PM, SANZ : LIST): LIST;
LINALGRN	: MGET(M, k, l : LIST): LIST;
SACLDIO	: MIAIM(A: LIST): LIST;
SACLDIO	: MICINS(A,V: LIST): LIST;
SACLDIO	: MICS(A: LIST): LIST;
SACM	: MIDCRA(M,ML,MLP,A,AL: LIST): LIST;
SACM	: MIDIF(M,A,B: LIST): LIST;
SACM	: MIEXP(M,A,N: LIST): LIST;
SACM	: MIHOM(M,A: LIST): LIST;
SACM	: MIINV(M,A: LIST): LIST;
SACM	: MINEG(M,A: LIST): LIST;
SACLDIO	: MINNCT(A: LIST): LIST;
CGBSYS	: MINPP(PP,PCO: LIST): LIST;
SACMPOL	: MIPDIF(RL,M,A,B: LIST): LIST;
SACMPOL	: MIPFSM(RL,M,A: LIST): LIST;
SACMPOL	: MIPHOM(RL,M,A: LIST): LIST;
SACMPOL	: MIPIPR(RL,M,D,A,B: LIST): LIST;
SACPFAC	: MIPISE(RL,M,D,A,B,S,T,C: LIST; VAR U,V: LIST);
SACMPOL	: MIPNEG(RL,M,A: LIST): LIST;
SACMPOL	: MIPPR(RL,M,A,B: LIST): LIST;
SACMPOL	: MIPRAN(RL,M,QL,N: LIST): LIST;
SACM	: MIPROD(M,A,B: LIST): LIST;
SACMPOL	: MIPSUM(RL,M,A,B: LIST): LIST;
SACM	: MIQ(M,A,B: LIST): LIST;
SACM	: MIRAN(M: LIST): LIST;

SACPGCD : MPUCPP(RL,PL,A: LIST; VAR AL,AB: LIST);
SACPGCD : MPUCS(RL,PL,A,CL: LIST): LIST;
SACMPOL : MPUP(RL,ML,CL,A: LIST): LIST;
SACPGCD : MPUPP(RL,PL,A: LIST): LIST;
SACMPOL : MPUQ(RL,PL,A,BL: LIST): LIST;
LINALGRN : MRANG(U: LIST): LIST;
SYZHLP : MREAD(VL : LIST): LIST;
LINALGRN : MSET(M, k, l, x : LIST): LIST;
SYZHLP : MTPLH(PM : LIST; VAR L : LIST): LIST;
SYZHLP : MTPLV(PM : LIST; VAR L : LIST): LIST;
LINALGRN : MTRANS(M : LIST): LIST;
SACMUFAC : MUPBQP(PL,A: LIST): LIST;
SACMUFAC : MUPDDF(PL,A: LIST): LIST;
SACMPOL : MUPDER(ML,A: LIST): LIST;
SACPGCD : MUPEGC(PL,A,B: LIST; VAR C,U,V: LIST);
SACMUFAC : MUPFBL(PL,A: LIST): LIST;
SACMUFAC : MUPFS(PL,A,B,DL: LIST): LIST;
SACPGCD : MUPGCD(PL,A,B: LIST): LIST;
SACPGCD : MUPHEG(PL,A,B: LIST; VAR C,V: LIST);
SACMPOL : MUPRAN(PL,NL: LIST): LIST;
SACEXT5 : MUPRC(PL,A,B: LIST; VAR C,RL: LIST);
SACPGCD : MUPRES(PL,A,B: LIST): LIST;
SACPGCD : MUPSFF(PL,A: LIST): LIST;
MODVAR : MVDeclareB(VAR var: BOOLEAN; name,comment: ARRAY OF CHAR;
MODVAR : MVDeclareL(VAR var: LIST; name,comment: ARRAY OF CHAR;
MODVAR : MVFLAG(sym:LIST): LIST;
MODVAR : MVGET(sym:LIST): LIST;
MODVAR : MVHLP(sym:LIST);
MODVAR : MVOFF(sym:LIST);
MODVAR : MVON(sym:LIST);
MODVAR : MVSET(sym,value:LIST);
MASU : MWRIT1(Y: LIST; top: BOOLEAN);
MASU : MWRITE(Y: LIST);
MASSYM2 : NAME(L: LIST): LIST;
SACSYM : NAME(L: LIST): LIST;
MASUGB : NEULF(STAP,DSUM,LSUM,I,V,PAR: LIST; VAR LFNEU,NEUST: LIST);
MASUGB : NEWDIF(L,D,DIFALT: LIST): LIST;
MASADOM : NewDom(S, s: ARRAY OF CHAR): Domain;
MASUGB : NEWL(LFTEMP,LFNEU,LFEND: LIST): LIST;
MASSTOR : NEWQUE(): LIST;
MASREP : NewRep(): LIST;
SYZHLP : NEXTPAIR(VAR P1, P2, PPL : LIST);
MASmtc : NextParm(VAR s: ARRAY OF CHAR): BOOLEAN;
CGBAPPL : NFEXEC(C,PPS,PP: LIST; VAR NOUT: LIST);
CGBSYS : NFORM(GA,FCO,P: LIST; VAR N0,N1: LIST);
CGBSYS : NFTOP(GA,FCO,P: LIST; VAR N0,N1: LIST);
CGBMAIN : NFWRIT(NOUT: LIST);
SYZFUNC : NLBGFUP(P1, P2, SP, SPN, SPFL, GB, SPAK, GBTM : LIST;
SYZFUNC : NLDGBRED(GB, GBTM : LIST; VAR SY, T : LIST): LIST;
SYZGB : NLGBE(PL, SANZ, L : LIST; VAR T : LIST): LIST;
SYZGB : NLGBEF(PL, SANZ, L : LIST; VAR GBTM, T : LIST): LIST;

SYZGB : NLGBF(PL, SANZ: LIST; VAR GBTM, T: LIST): LIST;
SYZMAIN : NLHEQ(PL, SANZ, SRD, T : LIST): LIST;
SYZMAIN : NLHSEQ(PM, SANZ, SRD, T : LIST): LIST;
SYZMAIN : NLIEQ(PL, P, SANZ, SRD, T : LIST): LIST;
SYZMAIN : NLISEQ(PM, PL, SANZ, SRD, T : LIST): LIST;
SYZGB : NLMGB(PM, SANZ : LIST; VAR T : LIST): LIST;
SYZFUNC : NLMMULT(SY1, GBTM : LIST; VAR T : LIST): LIST;
SYZFUNC : NLPLMULT(SY, PL : LIST; VAR T : LIST): LIST;
SYZFUNC : NLRCSP(GB, SPL : LIST; VAR T : LIST): LIST;
SYZFUNC : NLRCSPR(PL : LIST; VAR SP, T : LIST): LIST;
SYZFUNC : NLSPC(P1, P2 : LIST; VAR SPFL, SP, T : LIST);
SYZFUNC : NLSPCEGB(GB, L : LIST; VAR SPFL, SPL, T : LIST);
SYZFUNC : NLSPCGB(GB : LIST; VAR SPFL, SPL, T : LIST);
SYZFUNC : NLSYONP(GB1, GB2, GBTM : LIST; VAR T : LIST): LIST;
SYZHLP : NMREAD(VL, T : LIST): LIST;
NOETHER : NOEINF();
NOETHER : NOEL32(M, K: GAMMAINT): LIST;
NOETHER : NOENSP(PG: LIST);
NOETHER : NOEPIP(POL, FACT: LIST): LIST;
NOETHER : NOEPOW(PG: LIST; K: GAMMAINT): LIST;
NOETHER : NOEPPR(POL1, POL2, TERM: LIST): LIST;
NOETHER : NOEPRM(POL, TERM: LIST): LIST;
NOETHER : NOEPSM(POL1, POL2: LIST): LIST;
NOETHER : NOERED(PG, POL: LIST; VAR BASE, BASEPOL, REMPOL: LIST);
NOETHER : NOESRT(POL: LIST): LIST;
MASUGB : NONEWL(LFTEMP: LIST): LIST;
SYZHLP : NORMF(VAR PL, GBTM : LIST);
CGBAPPL : NSET(NN0,NN1,CC0,CC1: LIST; VAR C0,C1: LIST);
MASUGB : NULRNV(A: LIST): LIST;
CGBMAIN : NWRIT0(N: LIST);
CGBMAIN : NWRIT1(N: LIST);
MASO : OABS(R: LIST): LIST;
MASSYM : OCCURQ(AL,L: LIST): BOOLEAN;
MASO : OCOMP(R,S: LIST): LIST;
MASO : OCON(R: LIST): LIST;
MASO : ODIF(R,S: LIST): LIST;
MASO : ODREAD(): LIST;
MASO : ODWRITE(R,NL: LIST);
MASO : OEXP(A,NL: LIST): LIST;
MASO : OIM(R: LIST): LIST;
MASO : OINT(A: LIST): LIST;
MASO : ONEG(R: LIST): LIST;
MASO : ONINV(R: LIST): LIST;
MASO : ONREAD(): LIST;
MASO : ONWRITE(R: LIST);
MASO : OONE(R: LIST): LIST;
MASUGB : OPREAD(VAR PAR,OPT: LIST);
MASO : OPROD(R,S: LIST): LIST;
MASO : OQ(R,S: LIST): LIST;
MASO : ORAND(NL: LIST): LIST;
SACLIST : ORDER(AL: LIST): LIST;
MASO : ORE(R: LIST): LIST;

SACLIST : OREAD(): LIST;
SYZMAIN : OREC(P1, P2 : LIST; VAR P3, P4, T : LIST);
MASO : ORN(A: LIST): LIST;
MASO : ORNP(A, B: LIST): LIST;
MASBIOS : OStreamKind(): INTEGER;
MASO : OSUM(R,S: LIST): LIST;
MASBIOSU : OUT(A: LIST): LIST;
SACLIST : OWRITE(B: LIST);
MASSYM2 : PACK(L: LIST): LIST;
SACSYM : PACK(L: LIST): LIST;
SACLIST : PAIR(A,B: LIST): LIST;
MASPARSE : Parse(): LIST;
SACCOMB : PARTN(NL,P: LIST): LIST;
SACCOMB : PARTR(NL: LIST): LIST;
SACCOMB : PARTSS(PL: LIST): LIST;
DIPC : PBCLI(RL,A: LIST): LIST;
SACPOL : PBIN(AL1,EL1,AL2,EL2: LIST): LIST;
SACPOL : PCL(A: LIST): LIST;
MASUGB : PCOMP(X,Y: LIST): LIST;
SACEXT4 : PCONST(RL,A: LIST): LIST;
SACPOL : PDBORD(A: LIST): LIST;
CGBDSTR : PdCons(F,VD: LIST): LIST;
SACPOL : PDEG(A: LIST): LIST;
SACPOL : PDEGSV(RL,A,IL: LIST): LIST;
SACPOL : PDEGV(RL,A: LIST): LIST;
CGBDSTR : PdF(PD: LIST): LIST;
MASUGB : PDIF(R,S,DIFALT: LIST): LIST;
CGBDSTR : PdParts(PD: LIST; VAR F,VD: LIST);
SACPOL : PDPV(RL,A,IL,NL: LIST): LIST;
CGBDSTR : PdVd(PD: LIST): LIST;
CGBDSTR : PdWrite(PD: LIST);
SACEXT3 : PERMCY(P: LIST): LIST;
SACCOMB : PERMR(NL: LIST): LIST;
DIPC : PFDIP(A: LIST; VAR RL,B: LIST);
SACPOL : PFDP(RL,A: LIST): LIST;
CGBMISC : PFIDNOR(B, P: LIST): LIST;
CGBMISC : PFIGB(PFL, TF: LIST; VAR ONE: BOOLEAN): LIST;
CGBMISC : PFIGBA(PFL, P, TF: LIST; VAR ONE: BOOLEAN): LIST;
CGBMISC : PFILDNOR(B, P: LIST; VAR ZERO: BOOLEAN): LIST;
CGBMISC : PFILDS(B: LIST; VAR ONE: BOOLEAN): LIST;
CGBMISC : PFILNOR(B, P: LIST; VAR ZERO: BOOLEAN): LIST;
CGBMISC : PFILS(B: LIST; VAR ONE: BOOLEAN): LIST;
CGBMISC : PFINOR(B, P: LIST): LIST;
CGBMISC : PFLDIPL (DIPL, DOM, VARL: LIST): LIST;
CGBMISC : PFWRITE(P: LIST);
SACPOL : PINV(RL,A,KL: LIST): LIST;
MASUGB : PKEGEL(C,N,KALT: LIST): LIST;
SACPOL : PLBCF(RL,A: LIST): LIST;
SACPOL : PLDCF(A: LIST): LIST;
MASUGB : PLF(L,I,V,PAR,OPT: LIST);
DIPC : PLFDIL(A: LIST; VAR RL,B: LIST);
SYZHLP : PLHTP(PL : LIST): LIST;
SYZFUNC : PLMULT(SY, PL : LIST): LIST;

SYZHLP : PLVTM(PL, L : LIST): LIST;
SYZHLP : PLWR(PL, VL : LIST);
SACPOL : PMDEG(A: LIST): LIST;
SACPOL : PMON(AL,EL: LIST): LIST;
SACPOL : PMPMV(A,KL: LIST): LIST;
DIPC : PMPV(RL,A,IL,NL: LIST): LIST;
SYZHLP : PMWR(PM, VL : LIST);
SYZHLP : POL(PL, POS : LIST): LIST;
MASUGB : POLCOP(L: LIST): LIST;
SACPOL : PORD(A: LIST): LIST;
SYZHLP : POS(P, PL : LIST): LIST;
DIPE : POWSEV(PL,A: LIST): LIST;
DIPC : PPERMV(RL,A,P: LIST): LIST;
MASUGB : PREAD(VAR L,I,V: LIST);
SACPOL : PRED(A: LIST): LIST;
MASLISPU : PROCP(X: LIST): BOOLEAN;
MASUGB : PROJ(R,S,DIFALT,OLDL,I: LIST; VAR D,B,DEG: LIST);
MASUGB : PROJEC(R,S,DIFALT,OLDL,I,PAR: LIST): LIST;
CGBSYS : PRSCOP(PAIRS: LIST): LIST;
SACPOL : PRT(A: LIST): LIST;
SACEXT4 : PSDSV(RL,A,IL,NL: LIST): LIST;
SACPOL : PTBCF(RL,A: LIST): LIST;
DIPTOO : PTERM(RL,A: LIST): LIST;
DIPTOO : PTYP(RL,A: LIST): LIST;
SACPOL : PUFP(RL,A: LIST): LIST;
MASUGB : PUG(LF,I,V,P,DEGP,NURLF,PAR,LFQ: LIST): LIST;
MASUGB : PUGB(L,I,V,PAR,OPT: LIST);
SACEXT4 : PUNT(RL,A: LIST): LIST;
MASSYM2 : PUT(S,AL,A: LIST);
SACSYM : PUT(S,AL,A: LIST);
DIPTOO : PVDEMA(A: LIST): LIST;
MASCOMB : PVFLISTS(list1,list2:LIST):LIST;
MASQ : QABS(R: LIST): LIST;
MASQ : QCOMP(R,S: LIST): LIST;
MASQ : QCON(R: LIST): LIST;
MASQ : QDIF(R,S: LIST): LIST;
MASQ : QDREAD(): LIST;
MASQ : QDWRITE(R,NL: LIST);
MASQ : QEXP(A,NL: LIST): LIST;
MASQ : QIMi(R: LIST): LIST;
MASQ : QIMj(R: LIST): LIST;
MASQ : QIMk(R: LIST): LIST;
MASQ : QINT(A: LIST): LIST;
MASQ : QINV(R: LIST): LIST;
MASQ : QNEG(R: LIST): LIST;
MASQ : QNREAD(): LIST;
MASQ : QNWRITE(R: LIST);
MASQ : QONE(R: LIST): LIST;
MASQ : QPROD(R,S: LIST): LIST;
MASQ : QQ(R,S: LIST): LIST;
MASQ : QRAND(NL: LIST): LIST;
MASQ : QRE(R: LIST): LIST;
MASQ : QRN(A: LIST): LIST;

MASQ : QRN4(A, B, C, D: LIST): LIST;
MASQ : QSUM(R,S: LIST): LIST;
massig : raise(s: INTEGER): INTEGER;
SYZFUNC : RCSP(GB, SPL : LIST): LIST;
SYZFUNC : RCSPR(PL : LIST; VAR SP : LIST): LIST;
CGBSYS : RDNORM(COND,FCO,P: LIST; VAR NCO: LIST);
MASUGB : RDPAR(): LIST;
CGBDSTR : RDSYS(VD: LIST): LIST;
MASSTOR : RED(L: LIST): LIST;
SACLIST : RED2(L: LIST): LIST;
SACLIST : RED3(L: LIST): LIST;
SACLIST : RED4(L: LIST): LIST;
CGBFUNC : REDSRT(RALT,RNEU: LIST): LIST;
SACLIST : REDUCT(A,IL: LIST): LIST;
CGBSYS : REDUCT(PELEM: LIST): LIST;
CGBSYS : REFIND(PCO,P: LIST; VAR PLI,RE: LIST);
MASSYM2 : REMPRP(S,AL: LIST);
SACSYM : REMPRP(S,AL: LIST);
CGBSYS : REXTP(P: LIST): LIST;
ADTOOLS : RFDDADV(e: LIST; VAR rat, vl: LIST);
ADTOOLS : RFDDFIPDD(ipdd:LIST):LIST;
DIPRF : RFDEN(R: LIST): LIST;
DIPRF : RFDIF(R,S: LIST): LIST;
DIPRF : RFEXP(A,NL: LIST): LIST;
DIPRF : RFFIP(RL,A: LIST): LIST;
DIPRF : RFINV(R: LIST): LIST;
DIPRF : RFNEG(R: LIST): LIST;
DIPRF : RFNOV(R: LIST): LIST;
DIPRF : RFNUM(R: LIST): LIST;
DIPRF : RFONE(R: LIST): LIST;
DIPRF : RFPROD(R,S: LIST): LIST;
DIPRF : RFQ(R,S: LIST): LIST;
DIPRF : RFREAD(V: LIST): LIST;
DIPRF : RFRED(RL,A,B: LIST): LIST;
DIPRF : RFSIGN(R: LIST): LIST;
DIPRF : RFSUM(R,S: LIST): LIST;
DIPRF : RFWRIT(R,V: LIST);
SACROOT : RIB(RL,SL: LIST): LIST;
SACROOT : RILC(I,KL: LIST): LIST;
SACROOT : RINT(I: LIST): LIST;
SACRN : RIRNP(I,CL: LIST): LIST;
DIPROOT : RIRWRT(R,EPS: LIST);
CGBSYS : RMGRT(COND,PP: LIST): LIST;
SACRN : RNABS(R: LIST): LIST;
SACEXT2 : RNBCR(A,B: LIST; VAR M,N,KL: LIST);
SACRN : RNCEIL(RL: LIST): LIST;
SACRN : RNCOMP(R,S: LIST): LIST;
SACRN : RNDEN(R: LIST): LIST;
SACRN : RNDIF(R,S: LIST): LIST;
MASAPF : RNDRD(): LIST;
MASRN : RNDRD(): LIST;
MASRN : RNDWR(R,NL: LIST);
SACRN : RNDWR(R,NL: LIST);

LINALGRN : RNVSVSUM(A, B : LIST): LIST;
LINALGRN : RNVVPROD(A, B : LIST): LIST;
LINALGRN : RNVVSUM(A, B : LIST): LIST;
LINALGRN : RNVWRITE(A, s : LIST);
SACRN : RNWRIT(R: LIST);
DIPRNPOL : RPABS(RL,A: LIST): LIST;
SACEXT8 : RPAFME(RL,M,A,BL: LIST): LIST;
SACPGCD : RPBLGS(RL,A: LIST; VAR AL,BL,SL: LIST);
DIPRNPOL : RPCONST(RL,A: LIST): LIST;
SACRPOL : RPDIF(RL,A,B: LIST): LIST;
SACEXT4 : RPDMV(RL,A: LIST): LIST;
SACRPOL : RPEMV(RL,A,BL: LIST): LIST;
DIPRNPOL : RPEXP(RL,A,NL: LIST): LIST;
SACRPOL : RPFIP(RL,A: LIST): LIST;
SACRPOL : RPIMV(RL,A: LIST): LIST;
DIPRNPOL : RPLWRS(RL,A,V,S: LIST);
SACEXT4 : RPMAIP(RL,A: LIST): LIST;
DIPRNPOL : RPMON(RL,A: LIST): LIST;
SACRPOL : RPNEG(RL,A: LIST): LIST;
DIPRNPOL : RPONE(RL,A: LIST): LIST;
SACRPOL : RPPROD(RL,A,B: LIST): LIST;
SACRPOL : RPQR(RL,A,B: LIST; VAR Q,R: LIST);
SACRPOL : RPREAD(VAR RL,A,V: LIST);
SACRPOL : RPRNP(RL,AL,B: LIST): LIST;
DIPRNPOL : RPSIGN(RL,A: LIST): LIST;
SACRPOL : RPSUM(RL,A,B: LIST): LIST;
SACRPOL : RPWRIT(RL,A,V: LIST);
DIPRNPOL : RPWRTS(RL,A,V,S: LIST);
DIPRNPOL : RUPEGC(A,B: LIST; VAR C,U,V: LIST);
DIPRNPOL : RUPGCD(A,B: LIST): LIST;
DIPRNPOL : RUPHEG(A,B: LIST; VAR C,V: LIST);
DIPRNPOL : RUPLCM(A,B: LIST): LIST;
SACANF : RUPMRN(R: LIST): LIST;
MASUGB : SCMULT(I,U: LIST): LIST;
SACSET : SCOMP(AL,L: LIST): LIST;
CGBMAIN : SCOV(CONDA,CONS,B: LIST): LIST;
MASUGB : SCPROD(A,B: LIST): LIST;
SACSET : SDIFF(A,B: LIST): LIST;
SACCOMB : SDR(S: LIST; VAR A,I: LIST);
SACLIST : SECOND(L: LIST): LIST;
CGBMAIN : SEENR(AC: LIST; VAR NR: LIST);
MASUGB : SEENR(AC: LIST; VAR NR: LIST);
MASADOM : SetAbsFunc(d: Domain; f1: PROCF1);
MASSET : SetAdd(e,S:LIST):LIST;
MASSET : SetAddQ(e,S:LIST):LIST;
DIPDCGB : SetBranchProc(BP: INTEGER);
MASADOM : SetCnstFunc(d: Domain; f1: PROCF1B);
CGBFUNC : SETCOL(COND,COL: LIST): LIST;
MASADOM : SetCompFunc(d: Domain; f2: PROCF2);
MASSET : SetComplement(S1,S2:LIST):LIST;
MASSET : SetComplementQ(S1,S2:LIST):LIST;
MASADOM : SetConvFunc(d1, d2: Domain; f1: PROCF1);
DIPAGB : SetCPExtend(EXT: PROCF2);

MASREP : SetRep(n,e,r: LIST);
MASADOM : SetSignFunc(d: Domain; f1: PROCF1);
MASADOM : SetSumFunc(d: Domain; f2: PROCF2);
MASADOM : SetToipFunc(d: Domain; f1v1: PROCF1V1);
DIPDCGB : SetTraceLevel(TL: INTEGER);
CGBMISC : SetUnion(A,B: LIST): LIST;
MASSET : SetUnion(S1,S2:LIST):LIST;
MASSET : SetUnionQ(S1,S2:LIST):LIST;
DIPAGB : SetUpdate(UPD: PROCP1V2);
DIPDCGB : SetUpdateProc(UP: INTEGER);
DIPAGB : SetUpdateVariableWeight(UPD: PROCP0);
MASLISP : SETV(V, A: LIST; VAR ENV: LIST);
DIPDCGB : SetVarOrdOpt(VOO: INTEGER);
MASADOM : SetVlddFunc(d: Domain; f1: PROCF1);
MASADOM : SetWritFunc(d: Domain; p1: PROCP1);
MASLISP : SEXPRP(X: LIST): BOOLEAN;
SACCOMB : SFCS(A: LIST): LIST;
MASSTOR : SFIRST(L, a: LIST);
MASUGB : SFP(A,B: LIST): LIST;
MASBIOSU : SHUT(A: LIST): LIST;
SYZMAIN : SIC(PM, PL, SANZ, SRD : LIST): LIST;
massig : sigblock(mask: INTEGER):INTEGER;
massig : SigMask(s: INTEGER): INTEGER;
massig : signal(s: INTEGER; h: Action): Action;
MASLISPU : Signature(F: LIST; VAR PI, PO: LIST; VAR def: BOOLEAN);
massig : sigsetmask(mask: INTEGER): INTEGER;
MASSIGNAL : SigUsr1HandleDefault(signo: INTEGER);
MASBIOS : SILINE(VAR S, L, R : GAMMAINT);
MASF : SIN(A: LIST): LIST;
SYZMAIN : SINL(PM, PL, SANZ, SRD, T : LIST): LIST;
SACSET : SINTER(A,B: LIST): LIST;
MASBIOS : SIUNIT(S : ARRAY OF CHAR): GAMMAINT;
MASUGB : SKPRO2(A,B: LIST): LIST;
SACLIST : SLELT(A,IL,AL: LIST);
MASBIOS : SLIST(A : LIST; VAR S : ARRAY OF CHAR);
MASSYM2 : SMEMB(S,L: LIST): LIST;
SACSYM : SMEMB(S,L: LIST): LIST;
SACM : SMFMI(M,A: LIST): LIST;
SACMPOL : SMFMIP(RL,M,A: LIST): LIST;
MASBIOS : SOLINE(VAR S, L, R : GAMMAINT);
MASBIOS : SOUNIT(S : ARRAY OF CHAR): GAMMAINT;
SYZFUNC : SPC(P1, P2 : LIST; VAR SPFL, SP : LIST);
SYZFUNC : SPCEGB(GB, L : LIST; VAR SPFL, SPL : LIST);
SYZFUNC : SPCGB(GB : LIST; VAR SPFL, SPL : LIST);
MASLISP : SPECIALFORM(S: LIST): BOOLEAN;
CGBSYS : SPOL(COND,HA,HB: LIST): LIST;
MASF : SQRT(A: LIST): LIST;
MASSYM2 : SREAD(): LIST;
SACSYM : SREAD(): LIST;
MASSYM2 : SREAD1(): LIST;
SACSYM : SREAD1(): LIST;
MASSTOR : SRED(L, LP: LIST);
SACSYM2 : SSYTBAL;

SYSINFO : SysInfoWrite(s: SYSINFO);
SYZMAIN : SYTHC(SY, PM, VL : LIST);
SYZMAIN : SYTHNL(SY, PM, VL, T : LIST);
MASSYM2 : SYWRIT(S: LIST);
SACSYM : SYWRIT(S: LIST);
SYZHLP : TA(L : LIST; T : LIST): LIST;
MASBIOS : TAB(n : GAMMAINT);
MASLISP : TAG(V,T: LIST): LIST;
MASF : TAN(A: LIST): LIST;
MASUGB : TCOMP(X,Y: LIST): LIST;
CGBFUNC : TESTHT(COL: LIST):LIST;
MASYMDIP : TFDIRP(A, V: LIST): LIST;
SACLIST : THIRD(L: LIST): LIST;
MASSTOR : TIME(): GAMMAINT;
SYZHLP : TR(L : LIST; T : LIST): LIST;
MASYMDIP : TVARS(T: LIST): LIST;
MASLISP : TYPEOF(X: LIST): LIST;
MASLISP : TYPOFTAG(L: LIST): LIST;
MASUGB : UG(LF,I,V,STAP,P,NURLF,PAR: LIST): LIST;
MASUGB : UGB(L,I,V,PAR,OPT: LIST);
MASUGB : UGBBIN();
DIPE : UIPRES(A,B: LIST; VAR CL,KL: LIST);
DIPE : UIPRS1(A,B: LIST): LIST;
DIPE : UIPSIL(A,EL: LIST): LIST;
DIPE : UIPSIV(A,B: LIST): LIST;
MASSYM : UNIFY(A,B: LIST; VAR S: LIST): BOOLEAN;
LISTTOOLS : UPCASE(clist:LIST):LIST;
DIPAGB : UPDATE(h: LIST; VAR G,B: LIST);
DIPAGB : UpdateVariableWeight;
CGBMAIN : UPDCAS(ALIST,DALT,B: LIST): LIST;
CGBSYS : UPDPP(COND,P: LIST): LIST;
MASSYM : UREAD(): LIST;
MASSYM2 : UREAD(): LIST;
SACSYM : UREAD(): LIST;
SACSET : USCOMP(AL,L: LIST): LIST;
SACSET : USDIFF(A,B: LIST): LIST;
DIPDIM : USETCT(U,V: LIST): LIST;
SACSET : USINT(A,B: LIST): LIST;
SACSET : USUN(A,B: LIST): LIST;
MASSYM : UWRIT1(L: LIST);
MASSYM2 : UWRIT1(L: LIST);
SACSYM : UWRIT1(L: LIST);
MASSYM : UWRITE(L: LIST);
MASSYM2 : UWRITE(L: LIST);
SACSYM : UWRITE(L: LIST);
DIPTOOLS : ValisPop();
DIPTOOLS : ValisPush(valis: LIST);
CGBMISC : ValisReset();
CGBMISC : ValisSet(V: LIST);
MASLISP : VALOFTAG(L: LIST): LIST;
SACPOL : VCOMP(U,V: LIST): LIST;
CGBDSTR : VdCons(V,D: LIST): LIST;
CGBDSTR : VdD(Vd: LIST): LIST;

Chapter 17

Comment to Procedure and Module Index

about

InitExternalsG : Tell Modula and LISP about external compiled procedures.
InitExternalsS : Tell Modula and LISP about external compiled procedures.

absolute

ADABSF : Arbitrary domain absolute value.
APABS : Arbitrary precision floating point absolute value.
CABS : Complex number absolute value.
DIIPAB : Distributive integral polynomial absolute value.
DIRPAB : Distributive rational polynomial absolute value.
EIVABS : Exterior integral vector absolute value.
EIVAPP : Exterior integral vector absolute primitive part.
IABSF : Integer absolute value function.
IPABS : Integral polynomial absolute value.
MASABS : Absolute value.
OABS : Octonion number absolute value.
QABS : Quaternion number absolute value.
RNABS : Rational number absolute value.
RNVABS : Rational number list absolute values.
RPABS : Rational polynomial absolute value.
SetAbsFunc : Set absolute value function in domain.

actual

RQEPRRC : This global variable holds information over the actual polynomial ring

adaption

ALFA : Automatic Linear Form Adaption.
TA : T Adaption.

ALDES-2

Aparse : Parse a set of ALDES-2 declarations and algorithms.

algebra

DIPE : DIP Exterior Algebra Definition Module.
InitExternalsL : Initialize external compiled linear algebra procedures.
LINALG : Linear algebra definition module.
LINALGI : MAS Linear Algebra Integer Definition Module.
LINALGRN : MAS Linear Algebra Rational Number Definition Module.

algebraic

AFCOMP : Algebraic number field comparison.
AFDIF : Algebraic number field element difference.
AFFINT : Algebraic number field element from integer.
AFFRN : Algebraic number field element from rational number.
AFINV : Algebraic number field inverse.
AFNEG : Algebraic number field element negation.
AFPAFP : Algebraic number field polynomial algebraic number field element product.
AFPAFP : Algebraic number field polynomial algebraic number field element product.
AFPAFQ : Algebraic number field polynomial algebraic number field element quotient.
AFPAFQ : Algebraic number field polynomial algebraic number field element quotient.
AFPBRI : Algebraic number field polynomial basis real root isolation.
AFPCLL : Algebraic number field polynomial real root isolation, Collins-Loos
AFPDIF : Algebraic number field polynomial difference.
AFPDMV : Algebraic number field polynomial derivative, main variable.
AFPEMV : Algebraic number field polynomial evaluation of main variable.
AFPEV : Algebraic number field polynomial evaluation.
AFPFIP : Algebraic number field polynomial from integral polynomial.
AFPFRP : Algebraic number field polynomial from rational polynomial.
AFPINT : Algebraic number field polynomial integration.
AFPME : Algebraic number field, polynomial multiple evaluation.
AFPMON : Algebraic number field polynomial monic.
AFPMPR : Algebraic number field polynomial minimal polynomial of a real root.
AFPNEG : Algebraic number field polynomial negative.
AFPNIP : Algebraic number field polynomial normalize to integral polynomial.
AFPPR : Algebraic number field polynomial product.
AFPQR : Algebraic number field polynomial quotient and remainder.
AFPRCL : Algebraic number field polynomial real root isolation, Collins-Loos algorithm.
AFPRII : Algebraic number field polynomial real root isolation induction.
AFPRLS : Algebraic number field polynomial real root list separation.
AFPROD : Algebraic number field element product.
AFPRRI : Algebraic number field polynomial relative real root isolation.
AFPRRS : Algebraic number field polynomial real root separation.
AFPSUM : Algebraic number field polynomial sum.
AFQ : Algebraic number field quotient.
AFSIGN : Algebraic number field sign.

AFSUM : Algebraic number field element sum.
AFSUPB : Algebraic number field squarefree univariate polynomial squarefree
AFUPBA : Algebraic number field univariate polynomial squarefree basis
AFUPCB : Algebraic number field univariate polynomial coarsest squarefree basis.
AFUPGC : Algebraic number field univariate polynomial greatest common divisor
AFUPGS : Algebraic number field polynomial greatest squarefree divisor.
AFUPRB : Algebraic number field univariate polynomial root bound.
AFUPRL : Algebraic number field polynomial, root of a linear polynomial.
AFUPSF : Algebraic number field univariate polynomial squarefree factorization.
AFUPSR : Algebraic number field univariate polynomial, sign at a rational
ANDWR : Algebraic number decimal write.
ANFAF : Algebraic number from algebraic number field element.
ANFAF : Algebraic number from algebraic number field element.
ANIIPE : Algebraic number isolating interval for a primitive element.
ANPEDE : Algebraic number primitive element for a double extension.
ANREPE : Algebraic number represent element of a primitive extension.
APDWR : Algebraic point, decimal write.
DIGBSI : Distributive polynomial system algebraic number G basis sign.
DOMAF : MAS Domain Algebraic Number Definition Module.
DomLoadAF : Domain load algebraic number.
IPAFME : Integral polynomial, algebraic number field multiple evaluation.
RPAFME : Rational polynomial, algebraic number field multiple evaluation.
SACANF : SAC Algebraic Number Field Definition Module.

algorithm

AFPRCL : Algebraic number field polynomial real root isolation, Collins-Loos algorithm.
DIRPIB : Second Algorithm for computing the involutive Base for a given F.
IDEGCD : Integer doubly extended greatest common divisor algorithm.
IEGCD : Integer extended greatest common divisor algorithm.
IJACS : Integer Jacobi symbol algorithm.
IPCRA : Integral polynomial chinese remainder algorithm.
IPRICL : Integral polynomial real root isolation, Collins-Loos algorithm.
IPRODK : Integer product, Karatsuba algorithm.
LDSMKB : Linear diophantine system solution, modified Kannan and Bachem algorithm.
MAIPDE : Matrix of integral polynomials determinant, exact division algorithm.
MAIPDM : Matrix of integral polynomials determinant, modular algorithm.
MDCRA : Modular digit chinese remainder algorithm.
MDLCRA : Modular digit list chinese remainder algorithm.
MIDCRA : Modular integer digit chinese remainder algorithm.
MMPDMA : Matrix of modular polynomials determinant, modular algorithm.
MUPFBL : Modular univariate polynomial factorization-Berlekamp algorithm.
RNMINVI : Rational number matrix inversion, integer algorithm.

algorithms

Aparse : Parse a set of ALDES-2 declarations and algorithms.

all

ALLELN : UGB all linear forms from stack of projections.
ALLLF : UGB all linear forms from stack of projections and print.
LFALL : UGB all linear forms from stack of projections 1.

alphabetic

ACOMP : Alphabetic comparison.
ACOMP : Alphabetic comparison.
ACOMP1 : Alphabetic comparison, 1.
ACOMP1 : Alphabetic comparison, 1.

and

ADGCDC : Arbitrary domain greatest common divisor and cofactors.
ADGCDE : Arbitrary domain greatest common divisor and linear combination.
ADPCP : Arbitrary Domain polynomial content and primitive part.
ADPCPP : Arbitrary domain polynomial content and primitive part.
ADQR : Arbitrary domain quotient and remainder.
ADSMPROD : Arbitrary domain scalar and matrix product.
AFPQR : Algebraic number field polynomial quotient and remainder.
ALLLF : UGB all linear forms from stack of projections and print.
Aparse : Parse a set of ALDES-2 declarations and algorithms.
CdpCd : Case distinction and polynomial set case distinction part.
CdpCons : Case distinction and polynomial set construct.
CdpParts : Case distinction and polynomial set parts.
CdpPs : Case distinction and polynomial set polynomial set part.
CdpRead : Case distinction and polynomial set read.
CdpVd : Case distinction and polynomial set variable list and domain descriptor
CdpVd : Case distinction and polynomial set variable list and domain descriptor
CdpWrite : Case distinction and polynomial set write.
CGBOUT : Comprehensive-Groebner-Basis execute and output.
CgbVd : Comprehensive Groebner basis variable list and domain descriptor.
CompSummary : Compiled function and procedure summary.
DAND : Digit and.
DIDPALCMPC : Distributive domain polynomial array list check and mark polynomials.
DIIPALCMPC : Distributive integral polynomial array list check and mark polynomials.
DIIPCP : Distributive integral polynomial content and primitive part.
DIIPQR : Distributive integral polynomial quotient and remainder.
DILFPFL : Groebner bases and related procedures for recursive integral polynomials.
DIMIS : Dimension and maximal independent set.
DIPADS : Distributive polynomial advance and substitute.
DIPCPP : distributive polynomial content and primitive part.
DIPPCPP : distributive polynomial pseudo content and primitive part.
DIPQR : Distributive polynomial quotient and remainder.
DIPSSM : Distributive polynomial sort and select minimal.
DIPVOPP : Distributive polynomial variable ordering optimization and permutation

DIRPQR : Distributive rational polynomial quotient and remainder.
DQR : Digit quotient and remainder.
EIVCPP : Exterior integral vector content and primitive part.
EVDFSI : Exponent vector difference and sign.
FdCons : Formula and dimension construct.
FdD : Formula and dimension formula part.
FdF : Formula and dimension formula part.
FdParts : Formula and dimension parts.
FdV : Formula and dimension variable list part.
FdWrite : Formula and dimension write.
GBEF : Groebner Base with Exponent Vector Check and Factors.
GS1 : UGB generate stack of sorted polynomials and critical pairs 1.
GS2 : UGB generate stack of sorted polynomials and critical pairs 2.
GsVd : Groebner system variable list and domain descriptor.
IDQR : Integer-digit quotient and remainder.
IFCL2 : Integer, floor and ceiling, logarithm, base 2.
IGCDCF : Integer greatest common divisor and cofactors.
InitExternalsD : Initialize external compiled ideal decomposition and root procedures.
InitExternalsG : Tell Modula and LISP about external compiled procedures.
InitExternalsS : Tell Modula and LISP about external compiled procedures.
IPCPP : Integral polynomial content and primitive part.
IPGCDC : Integral polynomial greatest common divisor and cofactors.
IPICPP : Integral polynomial integer content and primitive part.
IPLCPP : Integral polynomial list of contents and primitive parts.
IPQR : Integral polynomial quotient and remainder.
IPSCPP : Integral polynomial sign, content, and primitive part.
IQR : Integer quotient and remainder.
ISMPROD : Integer scalar and matrix product.
IUPRC : Integral univariate polynomial resultant and cofactor.
IVSVPROD : Integer vector scalar and vector product.
IVSVSUM : Integer vector scalar and vector sum.
LDSMKB : Linear diophantine system solution, modified Kannan and Bachem algorithm.
MASLISPU : Types, S-Expresion Types and Indicators.
MASPGCD : MAS Polynomial GCD and RES System Definition Module.
MASQREM : Quotient and remainder.
MIUPQR : Modular integral univariate polynomial quotient and remainder.
MKLIST : UGB make trace and cuts.
MMPIQR : Modular monic polynomial mod ideal quotient and remainder.
MPGCDC : Modular polynomial greatest common divisor and cofactors.
MPQR : Modular polynomial quotient and remainder.
MPUCPP : Modular polynomial univariate content and primitive part.
MUPRC : Modular univariate polynomial resultant and cofactor.
NLGBEF : Non-Commutative Groebner Base with Exponent Vector Check and Factors.
NLSPCEGB : Non-Commutative S-Polynomials with Coefficients and Exponenetvector-Check
OPREAD : UGB options and parameter read.
Parse : Parse program and generate code.
PatternAStart : pattern and start.
PdF : Parametric dimension formula and dimension list part.
PdVd : Parametric dimension variable list and domain descriptor part.
RNFCL2 : Rational number floor and ceiling of logarithm, base 2.

RNSMPROD : Rational number scalar and matrix product.
RPBLGS : Rational polynomial base coefficients least common multiple, greatest common divisor, and sign.
RPQR : Rational polynomial quotient and remainder.
RRCSR : Real root count solve and reduce.
SACM : SAC Modular Digit and Integer Definition Module.
SACPGCD : SAC Polynomial GCD and RES System Definition Module.
SACPRIM : SAC Factorization and Prime Number Definition Module.
SetDIPAGBOptions : Set the trace flag, the strategy and the variable weight list of the
SetPCppFunc : Set Content and primitive part function.
SetQrFunc : Set quotient and remainder function in domain.
SPCEGB : S-Polynomials with Coefficients and Exponenetvector-Check for Groebner Base.
UGBBIN : UGB input, execute and output.
UPDATE : Update of extended ideal basis and extended critical pair list as required
VdCons : Variable list and domain descriptor construct.
VdD : Variable list and domain descriptor domain descriptor part.
VdParts : Variable list and domain descriptor parts.
VdRead : Variable list and domain descriptor read.
VdV : Variable list and domain descriptor variable list part.
VERIFY : Verify conditions and polynomials.
WPLIST : Write polynomials and pairs.
WriteDIPAGBOptions : Write the current trace flag, strategy and variable weight list of the

append

APP0 : Append 0.

applications

CGBAPPL : Comprehensive-Groebner-Bases Applications Definition Module.

apply

FORAPPLYAT : formula apply to atomic formular.
FORAPPLYATF2 : formula apply to atomic formula f2.
ForEachinList : For each element e in r apply function f.
ForEachinRep : For each pair (n,e) in r apply function f.
MLDAPPLYAT : maslog demonstration apply to atomic formulas.

arbirary

RRADCOUNT : Real root arbirary domain count.
RRUADCOUNT : Real root univariate arbirary domain count.

arbitraray

ADCAST : arbitraray domain cast.

arbitrary

AD2DIP : arbitrary domain to distributive polynomial.
ADABSF : Arbitrary domain absolute value.
ADCAN : Arbitrary Polynomial Cancel.
ADCHARPOL : Arbitrary domain characteristic polynomial.
ADCNST : Arbitrary domain constant test.
ADCOMP : Arbitrary domain comparison.
ADCONV : Arbitrary domain conversion.
ADDDFDIL : arbitrary domain domain descriptor from distributive polynomial list.
ADDDFDILD : arbitrary domain domain descriptor from distributive polynomial list
ADDDFDIP : arbitrary domain domain descriptor from distributive polynomial.
ADDDFDIPD : arbitrary domain domain descriptor from distributive polynomial or default.
ADDDFSTR : arbitrary domain domain descriptor from string.
ADDDREAD : Arbitrary domain, domain descriptor read.
ADDDWRIT : Arbitrary domain, domain descriptor write.
ADDIF : Arbitrary domain difference.
ADDNFDIL : arbitrary domain domain number from distributive polynomial list.
ADDNFDILD : arbitrary domain domain number from distributive polynomial list
ADDNFDIP : arbitrary domain domain number from distributive polynomial.
ADDNFDIPD : arbitrary domain domain number from distributive polynomial or default.
ADDNFEDIP : Arbitrary domain domain number from extended distributive polynomial.
ADEPCompose : Arbitrary domain extended polynomial compose.
ADEPCrumble : Arbitrary domain extended polynomial crumble.
ADEPdegree : Arbitrary domain extended polynomial degree.
ADEPheadterm : Arbitrary domain extended plynomial head-term.
ADEPleadingterm : Arbitrary polynomial leading term.
ADEPmark : Arbitrary domain extended polynomial mark.
ADEPNFJ : Arbitrary domain extended polynomial normalform in the sense of Janet.
ADEPpolynomial : Arbitrary domain extended polynomial polynomial.
ADEPtriple : Arbitrary domain extende polynomial triple.
ADEPuntriple : Arbitrary domain extended polynomial untriple.
ADEXP : Arbitrary domain exponentiation.
ADEXTRA : Arbitrary domain extra definition module.
ADFACT : Arbitrary domain factorization.
ADFACTO : Arbitrary domain factorization with variable order optimization.
ADFI : Arbitrary domain from integer.
ADFIP : Arbitrary domain from integral polynomial.
ADGCD : Arbitrary domain greatest common divisor.
ADGCDC : Arbitrary domain greatest common divisor and cofactors.
ADGCDE : Arbitrary domain greatest common divisor and linear combination.
ADGJredI : Arbitrary domain polynomial G Janet-reducible modulo I.
ADINV : Arbitrary domain inverse.
ADINVT : Arbitrary domain inverse existence test.
ADIredG : Arbitrary domain polynomial set I reducible modulo G.
ADLCM : Arbitrary domain least common multiple.
ADLGeqH : Arbitrary domain polynomial list G equal H.
ADLGinH : Arbitrary domain polynomial list G in H.
AdLoadConvFunc : arbitrary domain load conversion functions.
ADMPROD : Arbitrary domain matrix product.

ADMPTRACE : Arbitrary domain matrix product trace.
ADMSUM : Arbitrary domain matrix sum.
ADMTRACE : Arbitrary domain matrix trace.
ADMWRITE : Arbitrary domain matrix write.
ADNEG : Arbitrary domain negative.
ADONE : Arbitrary domain one.
ADPCP : Arbitrary Domain polynomial content and primitive part.
ADPCPP : Arbitrary domain polynomial content and primitive part.
ADPFACT : Arbitrary domain polynomial factorization.
ADPFDIP : arbitrary domain polynomial from distributive polynomial.
ADPFeqG : Arbitrary domain polynomial F equal G.
ADPIQ : Arbitrary domain polynomial integer quotient.
ADPNEG : Arbitrary domain polynomial negative.
ADPNF : Arbitrary domain polynomial normalform.
ADPNFJ : Arbitrary domain polynomial normalform in the sense of Janet.
ADPNFJS : Arbitrary domain polynomial normalform in the sense of Janet modulo a set
ADPROD : Arbitrary domain product.
ADPSFF : Arbitrary domain polynomial squarefree factorization.
ADPSP : Arbitrary domain polynomial S-polynomial.
ADPSUGNF : Arbitrary domain polynomial normal with sugar strategy normalform.
ADPSUGSP : Arbitrary domain polynomial normal with sugar strategy S-polynomial.
ADQR : Arbitrary domain quotient and remainder.
ADQUOT : Arbitrary domain quotient.
ADREAD : Arbitrary domain read.
ADREM : Arbitrary domain remainder.
ADRFFADIP : arbitrary domain rational function from arbitrary domain intgeral
ADRFFADIP : arbitrary domain rational function from arbitrary domain intgeral
ADRMDD : arbitrary domain remove domain descriptor informations.
ADSIG : Arbitrary domain signature.
ADSIGN : Arbitrary domain sign.
ADSMPROD : Arbitrary domain scalar and matrix product.
ADSUM : Arbitrary domain sum.
ADTOIP : Arbitrary domain to integral polynomial conversion.
ADTOOLS : Arbitrary Domain Tools Definition Module.
ADUM : Arbitrary domain unit matrix.
ADVSPROD : Arbitrary domain vector scalar product.
ADVSVPROD : Arbitrary domain vector scalar vector product.
ADVVSUM : Arbitrary domain vector vector sum.
ADVWRITE : Arbitrary domain vector write.
ADWRIT : Arbitrary domain write.
APABS : Arbitrary precision floating point absolute value.
APCMPR : Arbitrary precision floating point compare.
APCOMP : Arbitrary precision floating point composition.
APDIFF : Arbitrary precision floating point difference.
APEXP : Arbitrary precision floating point exponentiation.
APEXPT : Arbitrary precision floating point exponent.
APFINT : Arbitrary precision floating point from integer.
APFRN : Arbitrary precision floating point from rational number.
APLG10 : Arbitrary precision floating point logarithm base 10.
APMANT : Arbitrary precision floating point mantissa.
APNEG : Arbitrary precision floating point negative.

APNELD : Arbitrary precision floating point number of equal leading digits.
APPI : Arbitrary precision floating point pi.
APPROD : Arbitrary precision floating point product.
APQ : Arbitrary precision floating point quotient.
APROOT : Arbitrary precision floating point n-th root.
APSHFT : Arbitrary precision floating point shift.
APSIGN : Arbitrary precision floating point sign.
APSPRE : Arbitrary precision floating point set precision.
APSUM : Arbitrary precision floating point sum.
APWRIT : Arbitrary precision floating point write.
DIFPF : Distributive polynomial with arbitrary domain coefficients from
DILADNF : distributive polynomial list arbitrary domain normal form.
DILFPFL : Distributive polynomial list with arbitrary domain coefficients from
DIP2AD : distributive polynomial to arbitrary domain.
DIPADGB : distributive polynomial arbitrary domain groebner basis.
DIPADGBext : distributive polynomial arbitrary domain groebner basis extension.
DIPADGBunion : distributive polynomial arbitrary domain groebner basis union.
DIPADIRSET : distributive polynomial arbitrary domain irreducible set.
DIPADNF : distributive polynomial arbitrary domain normal form.
DIPADOM : DIP Arbitrary Domain Definition Module.
DIPAGB : DIP Arbitrary Domain Groebner Basis Definition Module.
DIPAGB : Distributive polynomial arbitrary domain Groebner basis.
DIPFADIP : distributive polynomial from arbitrary domain integral polynomial.
DIPONE : distributive polynomial arbitrary domain one.
DOMAPF : MAS Domain Arbitrary Precision Floating Point Definition Module.
DomLoadAPF : Domain load arbitrary precision floating point.
DomSummary : Arbitrary domain summary.
InitExternalsE : Initialize external compiled arbitrary domain procedures.
MASADOM : MAS Arbitrary Domain Definition Module.
MASAPF : MAS Arbitrary Precision Floating Point Definition Module.
RNFAP : Rational number from arbitrary precision floating point.
RRADNFORM : Real root arbitrary domain normal form.
RRADOM : Real Root Arbitrary Domain Definition Module.
RRADPOLMATRIX : Real root arbitrary domain polynomial matrix.
RRADQUADFORM : Real root arbitrary domain quadratic form.
RRADSTRCONST : Real root arbitrary domain structure constants.
RRADVARMATRICES : Real root arbitrary domain multiplication matrices of variables.
RRUADOM : Real Root Univariate Arbitrary Domain Definition Module.
RRUADPOLTOVEC : Real root univariate arbitrary domain polynomial to vector.
RRUADQUADFORM : Real root univariate arbitrary domain quadratic form.
RRUADSTRCONST : Real root univariate arbitrary domain structure constants.

arcus

ARCTAN : Arcus tangens.

argument

FORPBINOPA : formula parse binary operation argument.
FORPUNOPA : formula parse unary operation argument.
TfCtj : type formula coefficient tuples with joker argument.
TFGENJ : type formula generate with joker argument.

arguments

FORGARGS : formula get arguments.
FORPARGS : formula parse arguments.
FORPQUANTA : formula parse quantifier arguments.
FORPVARA : formula parse variable arguments.

arithmetic

AADV : Arithmetic advance.
InitExternalsA : Initialize external compiled arithmetic procedures.
InitExternalsJ : Initialize external compiled arithmetic procedures.
InitExternalsQ : Initialize external compiled arithmetic Q procedures.

array

ARRAYDEC : Generate array name declarations.
DIDPALCMPC : Distributive domain polynomial array list check and mark polynomials.
DIDPLEXTAL : Distributive domain polynomial list extend array list.
DIIPALCMPC : Distributive integral polynomial array list check and mark polynomials.
DIIPLEXTAL : Distributive integral polynomial list extend array list.
GENARRAY : Generate array reference symbol.

as

INITUPDATE : The initialization function as a first call of UPDATE.
UPDATE : Update of extended ideal basis and extended critical pair list as required

ascending

EVASC : Exponent vector ascending.

assignment

ASSPR : Assignment problem.

associate

ASSOC : Associate.
ASSOC : Associate.
ASSOCQ : Associate equal.
ASSOCQ : Associate equal.
RPMAIP : Rational polynomial monic associate of integral polynomial.

atom

AREAD : Atom read.
ATOM : Atom.
AWRITE : Atom write.
CLISTFA : character list from atom.

atomic

FORAPPLYAT : formula apply to atomic formular.
FORAPPLYATF2 : formula apply to atomic formula f2.
FORCOUNTAF : formula count atomic formulas.
FORISATOM : formula is atomic formula.
FORREPAFS : formula replace atomic formulas.
MLDAPPLYAT : maslog demonstration apply to atomic formulas.
MLDMKATOM : maslog demonstration make atomic formula.
pqatom : polynomial equation atomic formula.
PQBASE : symbol to mark a polynomial equation special atomic formula.
pqmkaf : polynomial equation simplification make atomic formula.
pqnegaf : negate atomic formula.
pqpaf : polynomial equation simplification parse atomic formula.
pqprtaf : polynomial equation simplification print atomic formula.
pqreadaf : polynomial equation read atomic formula.
pqsimplifyaf : polynomial equation simplify atomic formula.
pqsmart : polynomial equation atomic formula smart simplification.
pqtexwaf : polynomia equation tex write atomic formula.
tfmkaf : type formula make atomic formula.
tfpaf : type formula parse atomic formula.
tfshiftaf : type formula shift atomic formula.

attribute

ATTRIB : Attribute.
ATTRIB : Attribute.

augmentation

DIIGBA : Distributive integral polynomial groebner basis augmentation.
DIRGBA : Distributive rational polynomial groebner basis augmentation.
IPSFBA : Integral polynomial squarefree basis augmentation.
PFIGBA : Integral Polynomial Groebner Basis augmentation.

automatic

ALFA : Automatic Linear Form Adaption.
ALFRA : Automatic Linear Form Readaption.

axioms

DMIA : Define model, implementation or axioms.

bachem

LDSMKB : Linear diophantine system solution, modified Kannan and Bachem algorithm.

backspace

BACKUB : Backspace until blank.
BKSP : Backspace.

balance

SSYTBAL : System symbol tree balance.
STBAL : Symbol tree balance.
STBALS : Symbol tree balance subroutine.

base

APLG10 : Arbitrary precision floating point logarithm base 10.
BGFUP : Base Generators Factor Update.
DGBRED : Discrete Groebner Base Reduction.
DIDPELIMDGB : Distributive domain polynomial eliminate D-groebner base.
DIDPREDDGB : Distributive domain polynomial reduce D-groebner base.
DIGBZT : Distributive polynomial groebner base common zero test.
DIGFET : DIP G base successful extension test.
DIGISM : DIP G base index search for extension multiple univariats.
DIGISR : DIP G base index search for extension reductas.
DIIPELIMDGB : Distributive integral polynomial eliminate D-groebner base.
DIIPIB : Distributive integral involutive base.
DIIPIB2 : Distributive integral polynomial involutive base.
DIIPIB3 : Distributive integral polynomial involutive base.
DIIPREDDGB : Distributive integral polynomial reduce D-groebner base.
DIN1GB : Distributive non-commutative polynomials Groebner base.
DINCGB : Distributive non-commutative polynomials Groebner base.
DINLGB : Distributive non-commutative polynomials left Groebner base.
DINLGM : Distributive non-commutative minimal ordered left Groebner base.
DIPBCP : Distributive polynomial base coefficient product.
DIPIB3 : Distributive polynom involutive base.
DIPLBC : Distributive polynomial leading base coefficient.
DIPNBC : Distributive polynomial number of base coefficients.
DIPTBC : Distributive polynomial trailing base coefficient.
DIRGZS : Distributive rational Groebner base zero set.
DIRPIB : Second Algorithm for computing the involutive Base for a given F.
DLOG2 : Digit logarithm, base 2.
DNN2GB : distributive polynomials non-noetherian 2-sided Groebner base.
DNNLGB : distributive non-noetherian polynomials left Groebner base.
DNNRGB : distributive polynomials non-noetherian right Groebner base.
EVGBIT : Exponent vector groebner base intersection test.
FINDBC : Find base coefficient.
FLOG10 : Floating point logarithm base 10.
GBE : Groebner Base with Exponent Vector Check.
GBEF : Groebner Base with Exponent Vector Check and Factors.
GBF : Groebner Base with Factors.
GBZSET : Groebner base real zero set of zero dimensional ideal.
GINBAS : G-Symmetric Integral Base Construction.
GINCHKBAS : G-Symmetric Integral Base Check.
GRNBAS : G-Symmetric Rational Base Construction.
GRNCHKBAS : G-Symmetric Rational Base Check.
GRNGGB : G-Symmetric Rational Base Construction (Buchberger-Algorithm).
IBcrit : Inovlutive Base criterium.
IBeqGB : Inovlutive Base equal Groebner Base.
IBeqGB : Inovlutive Base equal Groebner Base.
IFCL2 : Integer, floor and ceiling, logarithm, base 2.
ILOG10 : Integer logarithm base 10.

ILOG2 : Integer logarithm, base 2.
InitDIPIIB : Init distributive integral involutive base.
MERGE : Noether Merge of Base Polynomials.
MGB : Modul Groebner Base.
MLOGBASE : Maslog Base Definition Module.
NLBGFUP : Non-Commutative Base Generators Factor Update.
NLDGBRED : Non-Commutative Discrete Groebner Base Reduction.
NLGBE : Non-Commutative Groebner Base with Exponent Vector Check.
NLGBEF : Non-Commutative Groebner Base with Exponent Vector Check and Factors.
NLGBF : Non-Commutative Groebner Base with Factors.
NLMGB : Non-Commutative Modul Groebner Base.
NLSPCGB : Non-Commutative S-Polynomials with Coefficients for Groebner Base.
NOENSP : Noether Number of Noetherian Base Polynomials.
PBCLI : Polynomial base coefficients list.
PLBCF : Polynomial leading base coefficient.
PQBASE : Polynomial Equation Base Definition Module.
PTBCF : Polynomial trailing base coefficient.
PUG : Universal Groebner base using precomputation.
PUGB : Universal Groebner base with precomputed linear forms.
RNFCL2 : Rational number floor and ceiling of logarithm, base 2.
RPBLGS : Rational polynomial base coefficients least common multiple, greatest common divisor, and sign.
SetDCIBDecomp : Set decompositional involutive base decomposition.
SetDCIBdepth : Set decompositional involutive base depth of tree.
SetDCIBopt : Set decompositional involutive base options.
SetDCIBTraceLevel : Set Decompositional involutive base Trace Level.
SetDCIBVarOrdOpt : Set decompositional involutive base variable order option.
SetDIPIBCancel : Set distributive polynomial involutive base cancel.
SetDIPIBCrit : Set distributive polynomial involutive base critera.
SetDIPIBISJ : Set distributive involutive base irreducible set in the sense of Janet.
SetDIPIBopt : Set distributive polynomial involutive base options.
SetDIPIBSelect : Set distributive polynomial involutive base select.
SPCEGB : S-Polynomials with Coefficients and Exponenetvector-Check for Groebner Base.
SPCGB : S-Polynomials with Coefficients for Groebner Base.
SYGB : Syzygy for Groebner Base.
SYGBE : Syzygy for Groebner Base with Exponent Vector.
SYZGB : Syzygy Groebner Base Definition Module.
TfUseDb : type formula use data base.
UG : Universal Groebner base.
UGB : Universal Groebner base.
UseDb : Use data base.
WRUGB : Write universal Groebner base.

based

LDSSBR : Linear diophantine system solution, based on Rosser ideas.
PQCnfSimplify : polynomial equation cnf based simplification.
PQDnfSimplify : polynomial equation dnf based simplification.

bases

DILFPFL : Groebner bases and related procedures for recursive integral polynomials.
DINNGB : DIP Groebner bases for non noetherian polynomial rings.
DIPDCGB : DIP Decompositional Groebner Bases Definition Module.
DIPDCIB : DIP Decompositional Involutive Bases Definition Module.
DIPDDGB : DIP Domain D-Groebner Bases Definition Module.
DIPGB : DIP Groebner Bases Definition Module.
DIPIDGB : DIP Integral D-Groebner Bases Definition Module.
DIPIGB : DIP Integral Groebner Bases Definition Module.
DIPRNGB : DIP Rational Groebner Bases Definition Module.
DIPSP : DIP Integral Function Groebner Bases Implementation Module.
GroebnerBases1 : Distributive polynomials decompositional groebner bases 1.
GroebnerBases2 : Distributive polynomials decompositional groebner bases 2.
IBLWR : Involutive bases list write.
InvolutiveBases : Involutive Bases.
MASNCGB : MAS Non-commutative Groebner Bases Definition Module.
MASUGB : Universal Groebner Bases Definition Module.
SetDCGBopt : Set options for decompositional groebner bases.
TIPRNGB : DIP Rational Extended Groebner Bases Definition Module.
WriteDCGBopt : write decompositional groebner bases options.

bases:

SetDecompProc : Set Decomposition-Procedure for decompositional groebner bases:
SetTraceLevel : Set Trace-Level for decompositional groebner bases:
SetUpdateProc : Set Update-Procedure for decompositional groebner bases:
SetVarOrdOpt : Set Variable-Order-Optimization for decompositional groebner bases:

basic

MASBIOS : MAS Basic I/O System Definition Module.
SACBIOS : SAC Basic I/O System Definition Module.

basis

AFPBRI : Algebraic number field polynomial basis real root isolation.
AFUPBA : Algebraic number field univariate polynomial squarefree basis
AFUPCB : Algebraic number field univariate polynomial coarsest squarefree basis.
CGBFGSYS : Comprehensive Groebner basis from Groebner system.
CGBFRM : Comprehensive-Groebner-Basis from coloured basis.
CGBGLOBRED : Comprehensive Groebner basis global reduce.
CgbI : Comprehensive Groebner basis number of conditions part.
CGBOPT : Comprehensive Groebner Basis Options.
CGBOPTWRITE : Comprehensive Groebner Basis Options Write
CgbP : Comprehensive Groebner basis polynomial list part.
CgbParts : Comprehensive Groebner basis parts.
CGBQFF : Comprehensive Groebner basis quantifier free formula.
CgbVd : Comprehensive Groebner basis variable list and domain descriptor.
CgbWrite : Comprehensive Groebner basis write.
DIDPDGB : Distributive domain polynomial D-groebner basis.
DIDPEGB : Distributive domain polynomial E-groebner basis.
DIGBC3 : Distributive polynomial groebner basis criterion 3.

berlekamp

beta

beta-integers

between

SwitchParse : Switch parsing between generic / non-generic parse.

binary

FORMKBINOP : formula make binary operation.
FORPBINOP : formula parse binary operation.
FORPBINOPA : formula parse binary operation argument.
IUPBEI : Integral univariate polynomial binary rational evaluation, integer output.
IUPBES : Integral univariate polynomial binary rational evaluation of sign.
IUPBHT : Integral univariate polynomial binary homothetic transformation.
IUPBRE : Integral univariate polynomial binary rational evaluation.
RNBCR : Rational number binary common representation.

binomial

IBCIND : Integer binomial coefficient induction.
IBCOEF : Integer binomial coefficient.
IBCPS : Integer binomial coefficient partial sum.
PBIN : Polynomial binomial.

BIOS

CloseBIOS : Close BIOS.
MASBIOSU : MAS BIOS Utility Definition Module.

bisection

RIB : Rational interval bisection.

bit

BITRAN : Bit, random.

black-box

FORISBBFOR : formula is black-box formula.
InitBbfParser : Initialize black-box formula parser.

blank

BACKUB : Backspace until blank.
BLINES : Blank lines.

blanks

CREADB : Character read, skipping blanks.

block

FORIMQB : formula innermost quantifier block.
sigblock : Block signals.

boolean

FORISBOOLVAR : formula is boolean variable.
MVDeclareB : modula variable declare boolean.

bound

AFUPRB : Algebraic number field univariate polynomial root bound.
FORCONTBDVAR : formula contain bound variable.
IPFCB : Integral polynomial factor coefficient bound.
IPGFCB : Integral polynomial Gelfond factor coefficient bound.
IUPRB : Integral univariate polynomial root bound.
MLDCONTBDVAR : maslog demonstration contains bound variable.

branch-Procedure

SetBranchProc : Set Branch-Procedure for procedure GroebnerBases2:

bubble

DILBBS : Distributive List Bubble Sort.
DILBSO : Distributive polynomial list bubble sort.
DILEBBS : Distributive List Extended Bubble Sort.
DIPBSO : Distributive polynomial bubble sort.
EVLRNBSO : Rational exponent vector list bubble sort.
LBIBS : List of beta-integers bubble sort.
LRNBS : List of rational numbers bubble sort.

bubble-merge

LBIBMS : List of beta-integers bubble-merge sort.
LRNBMS : List of rational numbers bubble-merge sort.

buffer

DIBUFF : Display input buffer.

calculation

DPCC : Digit partial cosequence calculation.
IPRCH : Integral polynomial real root calculation, high precision.
IPRCHS : Integral polynomial real root calculation, high-precision special.
IPRCN1 : Integral polynomial real root calculation, 1 root.
IPRCNP : Integral polynomial real root calculation, newton method preparation.

call

CallCompiled : Call compiled function or procedure.
DOS : Call DOS program.
INITUPDATE : The initialization function as a first call of UPDATE.

cancel

ADCAN : Arbitrary Polynomial Cancel.
DILCAN : Distributive Polynomial Cancel.
SetDIPIBCancel : Set distributive polynomial involutive base cancel.

cartesian

CPLEXN : Cartesian product, lexicographically next.

case

CDINIT : Case distinction init.
CdpCd : Case distinction and polynomial set case distinction part.
CdpCd : Case distinction and polynomial set case distinction part.
CdpCons : Case distinction and polynomial set construct.
CdpParts : Case distinction and polynomial set parts.
CdpPs : Case distinction and polynomial set polynomial set part.
CdpRead : Case distinction and polynomial set read.
CdpVd : Case distinction and polynomial set variable list and domain descriptor
CdpWrite : Case distinction and polynomial set write.
CdRead : Case distinction read.
CdWrite : Case distinction write.
CgbCd : Groebner system initial case distinction.
CONINI : Initialize case distinction.
GsCd : Groebner system initial case distinction.
STIC : Solution Test for inhomogenous commutative Case.
STINL : Solution Test for inhomogenous non-commutative Case.
SYTHC : Syzygy Test for homogenous commutative Case.
SYTHNL : Syzygy Test for homogenous non-commutative Case.
UPDCAS : Update case distinction.

cast

ADCAST : arbitraray domain cast.

ceiling

IFCL2 : Integer, floor and ceiling, logarithm, base 2.
RNCEIL : Rational number, ceiling of.
RNFCL2 : Rational number floor and ceiling of logarithm, base 2.

cells

CELLS : Cells.

center

DINCCP : Distributive rational non-commutative polynomial center polynomial.
DINCCPpre : Distributive rational non-commutative polynomial center polynomial preparation.
MASNCC : MAS Non-commutative Center Definition Module.

CGB

CGBLM : CGB coloured distributive polynomial list merge.
CGBLPM : CGB list merge.
SetUnion : Miscellaneous CGB Functions.

chain

NLRCSP : Non-Commutative Reduction Chain of S-Polynomials.
NLRCSPR : Reduction Chain of S-Polynomials with Remainder.
RCSP : Reduction Chain of S-Polynomials.
RCSPR : Reduction Chain of S-Polynomials with Remainder.

change

CHDOM : Change domain.

changes

TfSignChs : type formula sign changes.

character

CLIN : Character list in.
CLISTFA : character list from atom.
CLOUT : Character list out.
CLTIS : Character list to input stream.
CREAD : Character read.
CREADB : Character read, skipping blanks.
CWRIT2 : Character write, 2 characters.
CWRIT3 : Character write, 3 characters.
CWRIT4 : Character write, 4 characters.
CWRIT5 : Character write, 5 characters.
CWRIT6 : Character write, 6 characters.
CWRITE : Character write.
MASCHR : MAS character.
PACK : Pack character list.
PACK : Pack character list.
UPCASE : upcase character list.

characteristic

ADCHARPOL : Arbitrary domain characteristic polynomial.
CSFPAR : Characteristic set from partition.
CSINT : Characteristic set intersection.
CSSUB : Characteristic set subset.
CSUN : Characteristic set union.
ICHARPOL : Integral matrix characteristic polynomial.
SFCS : Set from characteristic set.

characters

CWRIT2 : Character write, 2 characters.
CWRIT3 : Character write, 3 characters.
CWRIT4 : Character write, 4 characters.
CWRIT5 : Character write, 5 characters.
CWRIT6 : Character write, 6 characters.

check

CHDEGL : Check degree of polynomial list.
DIDPALCMPC : Distributive domain polynomial array list check and mark polynomials.
DIIPALCMPC : Distributive integral polynomial array list check and mark polynomials.
GBE : Groebner Base with Exponent Vector Check.
GBEF : Groebner Base with Exponent Vector Check and Factors.
GINCHK : G-Symmetric Integral Polynomial Check.
GINCHKBAS : G-Symmetric Integral Base Check.
GRNCHK : G-Symmetric Rational Polynomial Check.
GRNCHKBAS : G-Symmetric Rational Base Check.
LFCHECK : Linear form check.
NLGBE : Non-Commutative Groebner Base with Exponent Vector Check.
NLGBEF : Non-Commutative Groebner Base with Exponent Vector Check and Factors.
SUBCHK : G-Symmetric Polynomial Check.
SUBSYM : G-symmetric Polynomial Symmetric Check.

chinese

IPCRA : Integral polynomial chinese remainder algorithm.
MDCRA : Modular digit chinese remainder algorithm.
MDLCRA : Modular digit list chinese remainder algorithm.
MIDCRA : Modular integer digit chinese remainder algorithm.

choice

IPCEVP : Integral polynomial, choice of evaluation points.

circle

IPVCHT : Integral polynomial variations after circle to half-plane transformation.
IUPCHT : Integral univariate polynomial circle to half-plane transformation.

class

DIPCLP : Distributiv Polynomial Class of Polynomial.
DIPCLT : Distributiv Polynomial Class of Term.

classification

Class2Sym : classification to symbol.
InitClassSyms : Initialize classification symbols.
Sym2Class : symbol to classification.

classify

TfClassify : type formula classify coefficient tuple.
TfClassifyI : type formula classify coefficient tuple interpreter version.

clock

clock : clock Foreign Module.

clocK

ClocK : ClocK returns milliseconds of the processes cpu-time.

clock

CLOCK : Clock.

close

CloseBIOS : Close BIOS.
CUNIT : Close unit.

cnf

FORMKCNF : formula make cnf.
PQCnfSimplify : polynomial equation cnf based simplification.

coarsest

AFUPCB : Algebraic number field univariate polynomial coarsest squarefree basis.
IPCSFB : Integral polynomial coarsest squarefree basis.

code

Parse : Parse program and generate code.

coefficient

DIPBCP : Distributive polynomial base coefficient product.
DIPCT : distributive polynomial coefficient tuple.
DIPLBC : Distributive polynomial leading base coefficient.
DIPLDC : Distributive polynomial leading coefficient.
DIPTBC : Distributive polynomial trailing base coefficient.
DIPTCF : Distributive polynomial trailing coefficient.
DIPTCS : Distributive polynomial trailing coefficient specified variable.
FINDBC : Find base coefficient.
IBCIND : Integer binomial coefficient induction.
IBCOEF : Integer binomial coefficient.
IBCPS : Integer binomial coefficient partial sum.
IPFCB : Integral polynomial factor coefficient bound.
IPGFCB : Integral polynomial Gelfond factor coefficient bound.
MCOEF : Make coefficient list.
PCL : Polynomial coefficient list.
PLBCF : Polynomial leading base coefficient.
PLDCF : Polynomial leading coefficient.
PTBCF : Polynomial trailing base coefficient.
TfClassify : type formula classify coefficient tuple.
TfClassifyI : type formula classify coefficient tuple interpreter version.
TfCtj : type formula coefficient tuples with joker argument.
TFFTUPLE : type formula from coefficient tuple with joker entries.

coefficients

DIFPF : Distributive polynomial with arbitrary domain coefficients from
DILFPFL : Distributive polynomial list with arbitrary domain coefficients from
DIPNBC : Distributive polynomial number of base coefficients.
IPPSC : Integral polynomial principal subresultant coefficients.
MCPMV : Matrix of coefficients of polynomials, with respect to main variable.
NLSPC : Non-Commutative S-Polynomial with Coefficients.
NLSPCEGB : Non-Commutative S-Polynomials with Coefficients and Exponenetvector-Check
NLSPCGB : Non-Commutative S-Polynomials with Coefficients for Groebner Base.
PBCLI : Polynomial base coefficients list.
RPBLGS : Rational polynomial base coefficients least common multiple, greatest common divisor, and sign.
SPC : S-Polynomial with Coefficients.
SPCEGB : S-Polynomials with Coefficients and Exponenetvector-Check for Groebner Base.
SPCGB : S-Polynomials with Coefficients for Groebner Base.

cofactor

IUPRC : Integral univariate polynomial resultant and cofactor.
MUPRC : Modular univariate polynomial resultant and cofactor.

cofactors

ADGCDC : Arbitrary domain greatest common divisor and cofactors.
IGCDCF : Integer greatest common divisor and cofactors.
IPGCDC : Integral polynomial greatest common divisor and cofactors.
MPGCDC : Modular polynomial greatest common divisor and cofactors.

collins-Loos

AFPCLL : Algebraic number field polynomial real root isolation, Collins-Loos
AFPRCL : Algebraic number field polynomial real root isolation, Collins-Loos algorithm.
IPRICL : Integral polynomial real root isolation, Collins-Loos algorithm.

colour

CMULT : Colour multiplication.
COLDIF : Colour difference.
COLPRD : Colour product.
INICOL : Initialize colour.
KEYCOL : Key colour.
MKACOL : Make colour.
MKCOL : Make new colour.
SETCOL : Set colour.
WRCOL : Write colour.

coloured

CGBCOL : Write coloured polynomials without green monomials.
CGBFRM : Comprehensive-Groebner-Basis from coloured basis.
CGBLM : CGB coloured distributive polynomial list merge.
ColpCol : Coloured polynomial colouring part.
ColpCons : Coloured polynomial construct.
ColpConsCond : Coloured polynomial construct from condition.
ColpHT : Coloured polynomial head term.
ColpIsCnst : Coloured polynomial is (non zero) constant.
ColpIsZero : Coloured polynomial is zero.
ColpParts : Coloured polynomial parts.
ColpPol : Coloured polynomial polynomial part.
DCLWR : Coloured polynomials list write.
EQPLCL : Equal lists of coloured polynomials.

colouring

ColCons : Colouring construct.
ColConsCond : Colouring construct from condition.
ColEmpty : Colouring empty.
ColHT : Colouring head term.
ColIsEmpty : Colouring is empty.
ColParts : Colouring parts.
ColpCol : Coloured polynomial colouring part.
ColRed : Colouring red.
ColWhite : Colouring white.
FINCOL : Finish colouring.

column

MDELCOL : Matrix delete column.
MICINS : Matrix of integers column insertion.
MICS : Matrix of integers column sort.
MINNCT : Matrix of integers, non-negative column transformation.

combination

ADGCDE : Arbitrary domain greatest common divisor and linear combination.
ILCOMB : Integer linear combination.
IVLC : Integer vector linear combination.
RNVLC : Rational number vector linear combination.
VILCOM : Vector of integers linear combination.

combinatorial

MASCOMB : MAS Combinatorial System Definition Module.

combinatorical

SACCOMB : SAC Combinatorical System Definition Module.

combine

IPFLC : Integral polynomial factor list combine.

common

ADGCD : Arbitrary domain greatest common divisor.
ADGCDC : Arbitrary domain greatest common divisor and cofactors.
ADGCDE : Arbitrary domain greatest common divisor and linear combination.
ADLCM : Arbitrary domain least common multiple.
AFUPGC : Algebraic number field univariate polynomial greatest common divisor
DEGCD : Digit extended greatest common divisor.
DGCD : Digit greatest common divisor.
DIGBZT : Distributive polynomial groebner base common zero test.
DIILFRCD : DIP integral list from DIP rational list using common denominator.
DIPC : DIP Common Polynomial System Definition Module.
DIPCJ : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPIB : DIP Common Polynomial System Definition Module in the sense of Janet.
ECPLCMHT : Extended critical pair select the least common multiple of head terms.
EVGCD : Exponent vector greatest common divisor.
EVLCM : Exponent vector least common multiple.
IDEGCD : Integer doubly extended greatest common divisor algorithm.
IEGCD : Integer extended greatest common divisor algorithm.
IGCD : Integer greatest common divisor.
IGCDCF : Integer greatest common divisor and cofactors.
IHEGCD : Integer half-extended greatest common divisor.
ILCM : Integer least common multiple.
IPGCDC : Integral polynomial greatest common divisor and cofactors.
IPLCM : Integral polynomial least common multiple.
MPGCDC : Modular polynomial greatest common divisor and cofactors.
MUPEGC : Modular univariate polynomial extended greatest common divisor.
MUPGCD : Modular univariate polynomial greatest common divisor.
MUPHEG : Modular univariate polynomial half-extended greatest common divisor.
RNBCR : Rational number binary common representation.
RPBLGS : Rational polynomial base coefficients least common multiple, greatest common divisor, and sign.
RPBLGS : Rational polynomial base coefficients least common multiple, greatest common divisor, and sign.
RUPEGC : Rational univariate polynomial extended greatest common divisor.
RUPGCD : Rational univariate polynomial greatest common divisor.
RUPHEG : Rational univariate polynomial half-extended greatest common divisor.
RUPLCM : Rational univariate polynomial least common multiple.

commutative

IEQ : Special Solution for inhomogenous commutative equation.
ISEQ : Special Solution for inhomogenous commutative system of equation.
SIC : Special Solution for inhomogenous commutative system of equation.
STIC : Solution Test for inhomogenous commutative Case.
SYHC : Syzygy for homogenous commutative system of equation.
SYTHC : Syzygy Test for homogenous commutative Case.

commutator

DINCCO : Distributive rational non-commutative polynomial commutator.

compare

APCMPR : Arbitrary precision floating point compare.
COMPA2 : UGB trace compare.
EVCOMP : Exponent vector compare.
EVCOMP : UGB exponent vector compare.
EVIGLC : Exponent vector inverse graded lexicographical compare.
EVILCI : Exponent vector inverse lexicographical compare inverse exponent vector.
EVILCP : Exponent vector inverse lexicographical compare.
EVITDC : Exponent vector inverse total degree compare.
EVLFCP : Exponent vector linear form compare.
EVLFCP : UGB exponent vector linear form compare.
EVRNC : Rational exponent vector compare.
EVRNGL : Rational exponent vector inverse graded lexicographical compare.
ILSCMP : Index list strong compare.
ILWCMP : Index list week compare.
LBLXCO : List of beta integers lexicographical compare.

comparison

ACOMP : Alphabetic comparison.
ACOMP : Alphabetic comparison.
ACOMP1 : Alphabetic comparison, 1.
ACOMP1 : Alphabetic comparison, 1.
ADCOMP : Arbitrary domain comparison.
AFCOMP : Algebraic number field comparison.
CCOMP : Complex number comparison.
FFCOMP : Finite field comparison.
ICOMP : Integer comparison.
OCOMP : Octonion number comparison.
QCOMP : Quaternion number comparison.
RILC : Rational interval length comparison.
RNCOMP : Rational number comparison.
SetCompFunc : Set comparison function in domain.
VCOMP : Vector comparison.

compiled

CallCompiled : Call compiled function or procedure.
Compiledf0 : Compiled function declaration f0.
Compiledf1 : Compiled function declaration f1.
Compiledf2 : Compiled function declaration f2.
Compiledf3 : Compiled function declaration f3.
Compiledf4 : Compiled function declaration f4.
Compiledf5 : Compiled function declaration f5.
Compiledp0 : Compiled function declaration p0.
Compiledp1 : Compiled function declaration p1.
Compiledp1v2 : Compiled function declaration p1v2.
Compiledp1v3 : Compiled function declaration p1v3.
Compiledp2 : Compiled function declaration p2.
Compiledp2v2 : Compiled function declaration p2v2.
Compiledp2v3 : Compiled function declaration p2v3.
Compiledp3 : Compiled function declaration p3.

Compiledp3v2 : Compiled function declaration p3v2.
Compiledp3v3 : Compiled function declaration p3v3.
Compiledp4 : Compiled function declaration p4.
Compiledp5 : Compiled function declaration p5.
CompSummary : Compiled function and procedure summary.
InitExternals : Initialize external compiled procedures.
InitExternalsA : Initialize external compiled arithmetic procedures.
InitExternalsB : Initialize external compiled polynomial procedures.
InitExternalsC : Initialize external compiled non-commutative polynomial procedures.
InitExternalsD : Initialize external compiled ideal decomposition and root procedures.
InitExternalsE : Initialize external compiled arbitrary domain procedures.
InitExternalsG : Tell Modula and LISP about external compiled procedures.
InitExternalsI : Initialize external compiled interface procedures.
InitExternalsJ : Initialize external compiled arithmetic procedures.
InitExternalsL : Initialize external compiled linear algebra procedures.
InitExternalsM : Initialize external compiled real root procedures.
InitExternalsML : Initialze external compiled logic procedures.
InitExternalsPQSMPL : initialize external compiled PQS-procedures.
InitExternalsQ : Initialize external compiled arithmetic Q procedures.
InitExternalsS : Tell Modula and LISP about external compiled procedures.
InitExternalsU : Initialize external compiled utility procedures.
Signature : Signature of a compiled function or procedure.

complement

SetComplement : Set complement.
SetComplementQ : set complement.
VVECC : variable vector complement.

complete

DIIPCOM : Distributive integral polynomial complete system.
DIPCOM : Distributive polynomial complete.
DIRPCOM : Distributive rational polynom complete system.

complex

CABS : Complex number absolute value.
CCOMP : Complex number comparison.
CCON : Complex number conjugate.
CDIF : Complex number difference.
CDREAD : Complex number decimal read.
CDWRITE : Complex number decimal write.
CEXP : Complex number exponentiation.
CIM : Complex number imaginary part.
CINT : Complex number from integer.
CNEG : Complex number negative.
CNINV : Complex number inverse.
CNREAD : Complex number read.
CNWRITE : Complex number write.
CONE : Complex number one.
CPROD : Complex number product.
CQ : Complex number quotient.
CRAND : Complex number, random.

CRE : Complex number real part.
CRN : Complex number from rational number.
CRNP : Complex number from pair of rational numbers.
CSUM : Complex number sum.
DOMC : MAS Domain Complex Number Definition Module.
DomLoadC : Domain load complex number.
MASC : MAS Complex Number Definition Module.

component

DO1 : UGB add last component to exponent vector.
EVCADD : Exponent vector component add.
EVCSUB : Exponent vector component subtract.
IKM : Integer vector component product.

compose

ADEPCompose : Arbitrary domain extended polynomial compose.

composition

APCOMP : Arbitrary precision floating point composition.
COMP : Composition.
COMP2 : Composition 2.
COMP3 : Composition 3.
COMP4 : Composition 4.
DIPCMP : Distributive polynomial composition.
DIPMC2 : UGB distributive polynomial composition 2.
DIPMCP : Distributive polynomial monomial composition.
INTDDCMP : integer domain descriptor composition.
IPDDCMP : integral polynomial domain descriptor composition.
IPDECMP : integral polynomial domain element composition.
PCOMP : UGB distributive polynomial composition.
SCOMP : Set composition.
USCOMP : Unordered set composition.

comprehensive

CGBFGSYS : Comprehensive Groebner basis from Groebner system.
CGBGLOBRED : Comprehensive Groebner basis global reduce.
CgbI : Comprehensive Groebner basis number of conditions part.
CGBOPT : Comprehensive Groebner Basis Options.
CGBOPTWRITE : Comprehensive Groebner Basis Options Write
CgbP : Comprehensive Groebner basis polynomial list part.
CgbParts : Comprehensive Groebner basis parts.
CGBQFF : Comprehensive Groebner basis quantifier free formula.
CgbVd : Comprehensive Groebner basis variable list and domain descriptor.
CgbWrite : Comprehensive Groebner basis write.

comprehensive-Groebner-Bases

CGBAPPL	: Comprehensive-Groebner-Bases Applications Definition Module.
CGBDSTR	: Comprehensive-Groebner-Bases Data-Structures Definition Module.
CGBFUNC	: Comprehensive-Groebner-Bases Utility Functions Definition Module.
CGBMAIN	: Comprehensive-Groebner-Bases Main Programms Definition Module.
CGBMISC	: Comprehensive-Groebner-Bases Miscellaneous Programs Definition Module.
CGBSYS	: Comprehensive-Groebner-Bases System Definition Module.

comprehensive-groebner-basis

ADDCGB	: Add polynomials to comprehensive-groebner-basis.

comprehensive-Groebner-Basis

CGBFRM	: Comprehensive-Groebner-Basis from coloured basis.
CGBIN	: Comprehensive-Groebner-Basis input.
CGBOUT	: Comprehensive-Groebner-Basis execute and output.
CPART	: Comprehensive-Groebner-Basis quantifier free formula.
MKCGB	: Make Comprehensive-Groebner-Basis.

comprehensive-groebner-basis

WRCGB	: Write comprehensive-groebner-basis.
WRRCGB	: Write reduced comprehensive-groebner-basis.

computation

NOEL32	: Noether SK Polynomial Computation.
NOEPOW	: Noether SK Power Sum Computation.
NOERED	: Noether G-Symmetric Polynomial Computation.
SUBPOW	: Noether SK Power Sum Computation for Substitution Groups.
SUBRED	: Noether G-Symmetric Polynomial Computation for Substitution Groups.

compute

CLF2	: UGB compute linear form from difference set 2.
CLF3	: UGB compute linear form from difference set 3.
COMPLF	: UGB compute linear form from difference set.
ComputeTypeFormula	: compute type formula.
MKSP1	: UGB compute next non-zero reduced S-polynomial.
NEULF	: UGB compute new linear forms from new terms.
TfComputeTf	: type formula compute type formulas.

computing

DIRPIB	: Second Algorithm for computing the involutive Base for a given F.

conc

TCOMP	: UGB list constructive conc.

concatenation

CCONC : Constructive concatenation.
CONC : Concatenation.
LCONC : List concatenation.

condition

ADDCON : Add to condition.
CCOVER : Cover condition.
ColConsCond : Colouring construct from condition.
ColpConsCond : Coloured polynomial construct from condition.
CondCons : Condition construct.
CondEmpty : Condition empty.
CondIsEmpty : Condition is empty.
CondNzero : Condition non-zero part.
CondParts : Condition parts.
CondPRead : Condition part read.
CondRead : Condition read.
CondWrite : Condition write.
CondZero : Condition zero part.
CPART : Condition part.
FormFCond : Formula from Condition.
OREC : Ore Condition.
SCOV : Search condition.

conditions

CgbI : Comprehensive Groebner basis number of conditions part.
CONDRD : Conditions read.
VERIFY : Verify conditions and polynomials.

configuration

MASCONF : MAS Configuration Definition Module.
MLMASLOG : MAS Logic Configuration Implementation Module.

conjugate

CCON : Complex number conjugate.
OCON : Octonion number conjugate.
QCON : Quaternion number conjugate.

conjunction

WRCONJ : Write conjunction.

constant

ADCNST	: Arbitrary domain constant test.
ColpIsCnst	: Coloured polynomial is (non zero) constant.
DIPCNST	: distributive polynomial is constant.
DIPCNSTR	: distributive polynomial constant relative to variables.
DIPXCM	: distributive polynomial extract constant monomials.
dummycnst	: Dummy constant.
EVCNSTR	: exponent vector constant relatively.
FORMKCNST	: formula make constant, i.
ILADDC	: Index list addition of constant.
PCONST	: Polynomial constant.
RPCONST	: Rational polynomial constant.
SetCnstFunc	: Set constant test function in domain.

constants

RRADSTRCONST	: Real root arbitrary domain structure constants.
RRISTRCONST	: Real root integral structure constants.
RRUADSTRCONST	: Real root univariate arbitrary domain structure constants.
RRUISTRCONST	: Real root univariate integral structure constants.

construct

CdpCons	: Case distinction and polynomial set construct.
CgbCons	: Groebner system construct.
ColCons	: Colouring construct.
ColConsCond	: Colouring construct from condition.
ColpCons	: Coloured polynomial construct.
ColpConsCond	: Coloured polynomial construct from condition.
CondCons	: Condition construct.
DIDPCPLMS1	: Distributive domain polynomial list construct pairs list merge sort.
DIDPLCPL4	: Distributive domain polynomial list construct pair list.
DIIPCPLMS1	: Distributive integral polynomial list construct pairs list merge sort.
DIIPLCPL4	: Distributive integral polynomial list construct pair list.
DILCPL	: Distributive polynomial list construct pair list.
DNLCPL	: distributive polynomial non-noetherian left construct pair list.
DNRCPL	: distributive polynomial non-noetherian right construct pair list.
FdCons	: Formula and dimension construct.
GsCons	: Groebner system construct.
PdCons	: Parametric dimension construct.
VdCons	: Variable list and domain descriptor construct.

constructed

DIDPUCPL1	: Distributive domain polynomial update constructed pairs list.
DIIPUCPL1	: Distributive polynomial D-update constructed pairs list.

construction

GINBAS	: G-Symmetric Integral Base Construction.
GRNBAS	: G-Symmetric Rational Base Construction.
GRNGGB	: G-Symmetric Rational Base Construction (Buchberger-Algorithm).
MUPBQP	: Modular univariate polynomial Berlekamp q polynomials construction.

constructive

CCONC : Constructive concatenation.
CINV : Constructive inverse.
SetMinusC : Set minus constructive.
SetMinusCQ : Set minus constructive equal.
TCOMP : UGB list constructive conc.

constructor

EDIPSUGCON : Extended distributive polynomial normal with sugar strategy constructor.

contain

FORCONTBDVAR : formula contain bound variable.
FORCONTVAR : formula contain variable.
MLDCONTVAR : maslog demonstration contain variable.

containment

DIRLCT : Distributive rational polynomial list ideal containment test.
USETCT : Unordered set containment test.

contains

MLDCONTBDVAR : maslog demonstration contains bound variable.

content

ADPCP : Arbitrary Domain polynomial content and primitive part.
ADPCPP : Arbitrary domain polynomial content and primitive part.
DIIPCP : Distributive integral polynomial content and primitive part.
DIPCPP : distributive polynomial content and primitive part.
DIPPCPP : distributive polynomial pseudo content and primitive part.
EIVCPP : Exterior integral vector content and primitive part.
IPC : Integral polynomial content.
IPCPP : Integral polynomial content and primitive part.
IPIC : Integral polynomial integer content.
IPICPP : Integral polynomial integer content and primitive part.
IPICS : Integral polynomial integer content subroutine.
IPSCPP : Integral polynomial sign, content, and primitive part.
MPUC : Modular polynomial univariate content.
MPUCPP : Modular polynomial univariate content and primitive part.
MPUCS : Modular polynomial univariate content subroutine.
SetPCppFunc : Set Content and primitive part function.

contents

getstck : Get contents of stack register.
gettoc : Get contents of toc register.
IPLCPP : Integral polynomial list of contents and primitive parts.

contract

ContractEt : contract vel.
ContractVel : contract vel.
PQCRELAND : contract relations or.
PQCRELOR : contract relations or.

conversion

ADCONV : Arbitrary domain conversion.
AdLoadConvFunc : arbitrary domain load conversion functions.
ADTOIP : Arbitrary domain to integral polynomial conversion.
DILCONV : distributive polynomial list conversion.
DIPCONV : distributive polynomial conversion.
FILWRITE : Polynomial conversion.
IIC : Isolating interval conversion.
SetConvFunc : Set conversion function in domain.

conversion-to-integer-polynomial

SetToipFunc : Set conversion-to-integer-polynomial function in domain.

copy

COPYOB : Copy object.
CopyRep : Copy representation.
COPYTOENV : Copy to environement.
POLCOP : Two level list copy.
PRSCOP : Pairs copy.

cosequence

DPCC : Digit partial cosequence calculation.

cosinus

COS : Cosinus.

count

FORCOUNTAF : formula count atomic formulas.
RQEPRRC : Real Quantifier Elimination with Parametric Real Root Count.
RRADCOUNT : Real root arbirary domain count.
RRCSR : Real root count solve and reduce.
RRICOUNT : Real root integral count.
RRUADCOUNT : Real root univariate arbirary domain count.
RRUICOUNT : Real root univariate integral count.
STCNT : Symbol table tree count.
STCNT : Symbol table tree count.
TfCount : type formula count.
TfCount1 : type formula count 1.

counter

CounterWR : Counter Write.
DecCounter : Decrement counter.
IncCounter : Increment counter.

cover

CCOVER : Cover condition.

cpu-time

ClocK : ClocK returns milliseconds of the processes cpu-time.

critera

SetDIPIBCrit : Set distributive polynomial involutive base critera.

criterion

DIGBC3 : Distributive polynomial groebner basis criterion 3.
DIGBC4 : Distributive polynomial groebner basis criterion 4.

criterium

IBcrit : Inovlutive Base criterium.

critical

CPEXTEND : Critical pair extend.
ECPINSERT : Extended critical pair insertion.
ECPLCMHT : Extended critical pair select the least common multiple of head terms.
ECPPOLY1 : Extended critical pair select the first extended distributive polynomial.
ECPPOLY2 : Extended critical pair select the second extended distributive
ECPSELECT : Select an extended critical pair from the extended critical pair list.
ECPSELECT : Select an extended critical pair from the extended critical pair list.
ECPSUGAR : Extended critical pair select sugar.
ECPUNEXTEND : Extended critical pair un-extend.
ECPWRITE : Extended critical pair write.
GS1 : UGB generate stack of sorted polynomials and critical pairs 1.
GS2 : UGB generate stack of sorted polynomials and critical pairs 2.
LECPUNEXTEND : List of extended critical pairs un-extend.
LECPWRITE : List of extended critical pairs write.
MKNEWP : UGB make new critical pairs.
MKPAIR : UGB make critical pairs for polynomial list.
SetCPExtend : Set the critical pair extension function.
SetDIPAGBStrategy : Set the DIPAGB strategy option for the extended critical pair selection.
SetECPInsert : Set the extended critical pair insertion function.
SetECPSelect : Set the extended critical pair selection procedure.
SetECPWrite : Set the extended critical pair write procedure.
UPDATE : Update of extended ideal basis and extended critical pair list as required
WriteDIPAGBStrategy : Write the DIPAGB strategy option for the extended critical pair selection

crumble

ADEPCrumble : Arbitrary domain extended polynomial crumble.

current

WriteDIPAGBOptions : Write the current trace flag, strategy and variable weight list of the

cut

GINCUT : G-Symmetric Integral Polynomial Cut.
GRNCUT : G-Symmetric Rational Polynomial Cut.

cuts

CUT : UGB set of cuts.
MKLIST : UGB make trace and cuts.

cyclic

PERMCY : Permutation, cyclic.

D-groebner

DIDPDGB : Distributive domain polynomial D-groebner basis.
DIDPELIMDGB : Distributive domain polynomial eliminate D-groebner base.
DIDPREDDGB : Distributive domain polynomial reduce D-groebner base.
DIIPDGB : Distributive integral polynomial D-groebner basis.
DIIPELIMDGB : Distributive integral polynomial eliminate D-groebner base.
DIIPREDDGB : Distributive integral polynomial reduce D-groebner base.

D-Groebner

DIPDDGB : DIP Domain D-Groebner Bases Definition Module.
DIPIDGB : DIP Integral D-Groebner Bases Definition Module.

d-irreducible

PFILDS : Integral polynomial list d-irreducible set.

D-normal

DIDPDNF : Distributive domain polynomial D-normal form.

D-Normal

PFILDNOR : Integral Polynomial List D-Normal Form.

D-update

DIIPUCPL1 : Distributive polynomial D-update constructed pairs list.

data

TfUseDb : type formula use data base.
UseDb : Use data base.

data-Structures

CGBDSTR : Comprehensive-Groebner-Bases Data-Structures Definition Module.

debug

DLSWRITE : debug level SWRITE.

decimal

ANDWR : Algebraic number decimal write.
APDWR : Algebraic point, decimal write.
CDREAD : Complex number decimal read.
CDWRITE : Complex number decimal write.
ODREAD : Octonion number decimal read.
ODWRITE : Octonion number decimal write.
QDREAD : Quaternion number decimal read.
QDWRITE : Quaternion number decimal write.
RNDRD : Rational number decimal read.
RNDRD : Rational number decimal read.
RNDWR : Rational number decimal write.
RNDWR : Rational number decimal write.
RNDWRS : Rational number decimal write special.

declaration

Compiledf0 : Compiled function declaration f0.
Compiledf1 : Compiled function declaration f1.
Compiledf2 : Compiled function declaration f2.
Compiledf3 : Compiled function declaration f3.
Compiledf4 : Compiled function declaration f4.
Compiledf5 : Compiled function declaration f5.
Compiledp0 : Compiled function declaration p0.
Compiledp1 : Compiled function declaration p1.
Compiledp1v2 : Compiled function declaration p1v2.
Compiledp1v3 : Compiled function declaration p1v3.
Compiledp2 : Compiled function declaration p2.
Compiledp2v2 : Compiled function declaration p2v2.
Compiledp2v3 : Compiled function declaration p2v3.
Compiledp3 : Compiled function declaration p3.
Compiledp3v2 : Compiled function declaration p3v2.
Compiledp3v3 : Compiled function declaration p3v3.
Compiledp4 : Compiled function declaration p4.
Compiledp5 : Compiled function declaration p5.

declarations

Aparse : Parse a set of ALDES-2 declarations and algorithms.
ARRAYDEC : Generate array name declarations.
MASBIOSU : Procedure declarations.
MASLISPU : Procedure declarations.
MASSYM : Procedure declarations.
MASYMDIP : Procedure declarations.

declare

Declare : Declare.
MVDeclareB : modula variable declare boolean.
MVDeclareL : modula variable declare list.

decode

DCENV : Decode environment.

decomposed

IMSDS : Integer matrix solve decomposed system.
RNMSDS : Rational number matrix solve decomposed system.

decomposition

DIPDEC0 : DIP Ideal Decomposition 0 System Definition Module.
DIRLPD : DIP rational polynomial ideal primary ideal decomposition.
DIRLPW : DIP rational polynomial ideal primary ideal decomposition write.
DIRPDA : DIP rational polynomial ideal primary ideal decomposition over Q(alpha).
InitExternalsD : Initialize external compiled ideal decomposition and root procedures.
PSDSV : Polynomial special decomposition, specified variable.
SetDCIBDecomp : Set decompositional involutive base decomposition.

decomposition-Procedure

SetDecompProc : Set Decomposition-Procedure for decompositional groebner bases:

decompositional

DIPDCGB : DIP Decompositional Groebner Bases Definition Module.
DIPDCIB : DIP Decompositional Involutive Bases Definition Module.
GroebnerBases1 : Distributive polynomials decompositional groebner bases 1.
GroebnerBases2 : Distributive polynomials decompositional groebner bases 2.
SetDCGBopt : Set options for decompositional groebner bases.
SetDCIBDecomp : Set decompositional involutive base decomposition.
SetDCIBdepth : Set decompositional involutive base depth of tree.
SetDCIBopt : Set decompositional involutive base options.
SetDCIBTraceLevel : Set Decompositional involutive base Trace Level.
SetDCIBVarOrdOpt : Set decompositional involutive base variable order option.
SetDecompProc : Set Decomposition-Procedure for decompositional groebner bases:
SetTraceLevel : Set Trace-Level for decompositional groebner bases:
SetUpdateProc : Set Update-Procedure for decompositional groebner bases:
SetVarOrdOpt : Set Variable-Order-Optimization for decompositional groebner bases:
WriteDCGBopt : write decompositional groebner bases options.

decrement

DecCounter : Decrement counter.

default

ADDDFDIPD : arbitrary domain domain descriptor from distributive polynomial or default.
ADDNFDIPD : arbitrary domain domain number from distributive polynomial or default.
SigUsr1HandleDefault : SIGUSR1 default signal handler.

define

DEFE : Define expr function.
DEFF : Define fexpr function.
DEFM : Define macro function.
DEFMAP : Define generic map function.
DEFPROC : Define generic proc function.
DEFRULE : Define generic rule function.
DIPTODEF : DIP define distributive polynomial term order.
DIPVDEF : DIP define distributive polynomial variable list.
DMIA : Define model, implementation or axioms.
DSPEC : Define specification.

definition

ADEXTRA : Arbitrary domain extra definition module.
ADTOOLS : Arbitrary Domain Tools Definition Module.
ALDPARSE : Aldes Parser Definition Module.
CGBAPPL : Comprehensive-Groebner-Bases Applications Definition Module.
CGBDSTR : Comprehensive-Groebner-Bases Data-Structures Definition Module.
CGBFUNC : Comprehensive-Groebner-Bases Utility Functions Definition Module.
CGBMAIN : Comprehensive-Groebner-Bases Main Programms Definition Module.
CGBMISC : Comprehensive-Groebner-Bases Miscellaneous Programs Definition Module.
CGBSYS : Comprehensive-Groebner-Bases System Definition Module.
DIPADOM : DIP Arbitrary Domain Definition Module.
DIPAGB : DIP Arbitrary Domain Groebner Basis Definition Module.
DIPC : DIP Common Polynomial System Definition Module.
DIPCJ : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPDCGB : DIP Decompositional Groebner Bases Definition Module.
DIPDCIB : DIP Decompositional Involutive Bases Definition Module.
DIPDDGB : DIP Domain D-Groebner Bases Definition Module.
DIPDEC0 : DIP Ideal Decomposition 0 System Definition Module.
DIPDIM : DIP Dimension Definition Module.
DIPE : DIP Exterior Algebra Definition Module.
DIPGB : DIP Groebner Bases Definition Module.
DIPGCD : DIP GCD Definition Module.
DIPI : DIP Integral Definition Module.
DIPIB : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPIDEAL : DIP Ideal System Definition Module.
DIPIDGB : DIP Integral D-Groebner Bases Definition Module.
DIPIGB : DIP Integral Groebner Bases Definition Module.
DIPIIB : DIP Integral Polynomial System Definition Module in the sense of Janet.
DIPIPOL : DIP Integer Polynomial Definition Module.
DIPRF : DIP Rational Function Definition Module.
DIPRN : DIP Rational Definition Module.
DIPRNGB : DIP Rational Groebner Bases Definition Module.
DIPRNIB : DIP Rational Numbers Polynomial Definition Module in the sense of Janet.
DIPRNPOL : DIP Rational Number Polynomial Definition Module.
DIPROOT : DIP Ideal Real Root System Definition Module.

DIPTOO : DIP Termorder Optimization Definition Module.
DIPTOOLS : Distributive Polynomials Tools Definition Module.
DIPZ : DIP Zero Dimensional Ideal Definition Module.
DOMAF : MAS Domain Algebraic Number Definition Module.
DOMAPF : MAS Domain Arbitrary Precision Floating Point Definition Module.
DOMC : MAS Domain Complex Number Definition Module.
DOMFF : MAS Domain Finite Field Definition Module.
DOMI : MAS Domain Integer Definition Module.
DOMIP : MAS Domain Integral Polynomial Definition Module.
DOMMD : MAS Domain Modular Digit Definition Module.
DOMMI : MAS Domain Modular Integer Definition Module.
DOMO : MAS Domain Octonion Number Definition Module.
DOMQ : MAS Domain Quaternion Number Definition Module.
DOMRF : MAS Domain Rational Function Definition Module.
DOMRN : MAS Domain Rational Number Definition Module.
DOMRP : MAS Domain Rational Polynomial Definition Module.
GSYMFUIN : G-Symmetric Integral Polynomial System Definition Module.
GSYMFURN : G-Symmetric Rational Polynomial System Definition Module.
GSYMINP : GSYM Input Definition Module.
LINALG : Linear algebra definition module.
LINALGI : MAS Linear Algebra Integer Definition Module.
LINALGRN : MAS Linear Algebra Rational Number Definition Module.
LISTTOOLS : List Tools Definition Module.
MASADOM : MAS Arbitrary Domain Definition Module.
MASAPF : MAS Arbitrary Precision Floating Point Definition Module.
MASBIOS : MAS Basic I/O System Definition Module.
MASBIOSU : MAS BIOS Utility Definition Module.
MASC : MAS Complex Number Definition Module.
MASCOMB : MAS Combinatorial System Definition Module.
MASCONF : MAS Configuration Definition Module.
MASELEM : MAS Elementary Functions Definition Module.
MASERR : MAS Error Definition Module.
MASF : MAS Floating Point Definition Module.
MASFF : MAS Finite Field Definition Module.
MASI : MAS Integer Definition Module.
MASLISP : MAS Lisp Definition Module.
MASLISPU : MAS Lisp Utility Definition Module.
MASLOAD : MAS Load Definition Module.
MASLOADA : MAS Load Definition Module A.
MASLOADB : MAS Load Definition Module B.
MASLOADC : MAS Load Definition Module C.
MASLOADD : MAS Load Definition Module D.
MASLOADE : MAS Load Definition Module E.
MASLOADG : MAS Load Symmetric Functions Definition Module.
MASLOADJ : MAS Load Definition Module J.
MASLOADL : MAS Load Definition Module L.
MASLOADM : MAS Load Definition Module M.
MASLOADQ : MAS Load Definition Module Q.
MASLOADS : MAS Load Syzygy Definition Module.
MASLOG : Maslog Definition Module.
MASmtc : MAS mtc [Modula-2 to C] Definition Module.
MASNC : MAS Non-commutative Product Definition Module.
MASNCC : MAS Non-commutative Center Definition Module.

MASNCGB : MAS Non-commutative Groebner Bases Definition Module.
MASO : MAS Octonion Number Definition Module.
MASPARSE : MAS Parser Definition Module.
MASPGCD : MAS Polynomial GCD and RES System Definition Module.
MASQ : MAS Quaternion Number Definition Module.
MASREP : MAS Representation Definition Module.
MASRN : MAS Rational Number Definition Module.
MASSET : MAS Set Definition Module.
MASSIGNAL : MAS Signal Handling Definition Module.
MASSPEC : MAS Specification Definition Module.
MASSTOR : MAS Storage Definition Module.
MASSYM : MAS Symbol Definition Module.
MASSYM2 : MAS/SAC Symbol System Definition Module 2.
MASU : MAS Utility Definition Module.
MASUGB : Universal Groebner Bases Definition Module.
MASYMDIP : MAS Symbol to DIP Definition Module.
MLDEMO : Maslog Demonstration Definition Module.
MLOGBASE : Maslog Base Definition Module.
MLOGIO : Maslog Input Output System Definition Module.
MLPQSMPL : Masload Polynomial Equation Simplify Definition Module.
NOETHER : Noether Polynomial System Definition Module.
Portab : Portability Definition Module.
PQBASE : Polynomial Equation Base Definition Module.
PQSMPL : Polynomial Equation Simplification Definition Module.
RRADOM : Real Root Arbitrary Domain Definition Module.
RRINT : Real Root Integral Definition Module.
RRUADOM : Real Root Univariate Arbitrary Domain Definition Module.
RRUINT : Real Root Univariate Integral Definition Module.
SACANF : SAC Algebraic Number Field Definition Module.
SACBIOS : SAC Basic I/O System Definition Module.
SACCOMB : SAC Combinatorical System Definition Module.
SACD : SAC Digit Definition Module.
SACDPOL : SAC Dense Polynomial Definition Module.
SACEXT1 : SAC Extensions 1 Definition Module.
SACEXT2 : SAC Extensions 2 Definition Module.
SACEXT3 : SAC Extensions 3 Definition Module.
SACEXT4 : SAC Extensions 4 Definition Module.
SACEXT5 : SAC Extensions 5 Definition Module.
SACEXT6 : SAC Extensions 6 Definition Module.
SACEXT7 : SAC Extensions 7 Definition Module.
SACEXT8 : SAC Extensions 8 Definition Module.
SACI : SAC Integer Definition Module.
SACIPOL : SAC Integer Polynomial System Definition Module.
SACLDIO : SAC Linear Diophantine Equation System Definition Module.
SACLIST : SAC List Processing Definition Module.
SACM : SAC Modular Digit and Integer Definition Module.
SACMPOL : SAC Modular Polynomial Definition Module.
SACMUFAC : SAC Modular Univariate Polynomial Factorization Definition Module.
SACPFAC : SAC Polynomial Factorization Definition Module.
SACPGCD : SAC Polynomial GCD and RES System Definition Module.
SACPOL : SAC Polynomial System Definition Module.
SACPRIM : SAC Factorization and Prime Number Definition Module.
SACRN : SAC Rational Number Definition Module.

SACROOT : SAC Polynomial Real Root Definition Module.
SACRPOL : SAC Rational Polynomial Definition Module.
SACSET : SAC Set Definition Module.
SACSYM : SAC Symbol System Definition Module.
SACSYM2 : SAC Symbol 2 Definition Module.
SACUPFAC : SAC Univariate Polynomial Factorization Definition Module.
SUBST : Substitution Group Polynomial System Definition Module.
SYMMFU : Symmetric Functions Definition Module.
SYSINFO : System Informations Definition Module.
SYZFUNC : Syzygy Functions Definition Module.
SYZGB : Syzygy Groebner Base Definition Module.
SYZHLP : Syzygy Utility Programs Definition Module.
SYZMAIN : Syzygy Main Programs Definition Module.
TFORM : Type Formula Definition Module.
TIPRNGB : DIP Rational Extended Groebner Bases Definition Module.

degree

ADDTDG : Add total degree.
ADEPdegree : Arbitrary domain extended polynomial degree.
ADVTDG : Advance total degree.
CHDEGL : Check degree of polynomial list.
DEGRE : UGB total degree of a list of rational exponent vectors.
DILATDG : Distributive polynom list add total degree.
DILTDG : Distributive polynomial list total degree
DIPDEG : Distributive polynomial degree.
DIPDEGI : distributive polynomial degree of i-th main variable.
DIPDEM : Distributive polynomial degree matrix.
DIPDEV : Distributive polynomial degree vector.
DIPLDM : Distributive polynomial list degree matrix.
DIPLMD : distributive polynomial list maximum degree.
DIPRWTDG : Distributive polynomial rational-weighted total degree.
DIPTDG : Distributive polynomial total degree.
DMEVAD : Degree matrix exponent vector add.
EVITDC : Exponent vector inverse total degree compare.
EVLGTD : Exponent vector list generate for total degree.
EVRWTDEG : Exponent vector rational-weighted total degree.
EVTDEG : Exponent vector total degree.
IUPFDS : Integral univariate polynomial factor degree set.
LDEG : Distributive polynomial list total degree.
MUPDDF : Modular univariate polynomial distinct degree factorization.
PDEG : Polynomial degree.
PDEGSV : Polynomial degree, specified variable.
PDEGV : Polynomial degree vector.
PMDEG : Polynomial modified degree.
PVDEMA : Permutation vector for degree matrix.

delete

EVDEL : Exponent vector delete.
MDELCOL : Matrix delete column.
VDELEL : Vector delete element.

demonstration

InitExternalsMLDEMO : Initialize externals maslog demonstration procedures.
MLDAPPLYAT : maslog demonstration apply to atomic formulas.
MLDCONTBDVAR : maslog demonstration contains bound variable.
MLDCONTVAR : maslog demonstration contain variable.
MLDEMO : Maslog Demonstration Definition Module.
MLDIREAD : maslog demonstration infix read.
MLDMKATOM : maslog demonstration make atomic formula.
MLDMKCNF : maslog demonstration make disjunctive normal form.
MLDMKDNF : maslog demonstration make disjunctive normal form.
MLDMKPOS : maslog demonstration make positive.
MLDMKPOS1 : maslog demonstration make positive 1.
MLDMKPRENEX : maslog demonstration make prenex.
MLDMKVD : maslog demonstration make variables disjoint.
MLDPPRT : maslog demonstration pretty print.
MLDPREAD : maslog demonstration prefix read.
MLDPREPQE : maslog demonstration prepare quantifier elimination.
MLDSMPL : maslog demonstration simplify.
MLDSUBSTVAR : maslog demonstration substitute variables.
MLDTEXW : maslog demonstration tex write.
MLDTST : maslog demonstration test 1.
MLMLDEMO : MAS Logic Demonstration Implementation Module.
PQDEMO : Demonstration for this package.

denominator

DIILFRCD : DIP integral list from DIP rational list using common denominator.
RFDEN : Rational function denominator.
RNDEN : Rational number denominator.

dense

DMPPRD : Dense modular polynomial product.
DMPSUM : Dense modular polynomial sum.
DMUPNR : Dense modular univariate polynomial natural remainder.
DPFP : Dense polynomial from polynomial.
PFDP : Polynomial from dense polynomial.
SACDPOL : SAC Dense Polynomial Definition Module.

dependency

DIPLDV : Distributive polynomial list dependency on variables.
EVDOV : Exponent vector dependency on variables.

depth

SetDCIBdepth : Set decompositional involutive base depth of tree.

dequeue

DEQUE : Dequeue.

derivation

DIIPDM : Distributive integral polynomial derivation main variable.
DIIPDR : Distributive integral polynomial derivation.
DIIPHD : Distributive integral polynomial higher derivation.
DIMPAD : distributive monomial partial derivation.
DIPPAD : distributive polynomial partial derivation.
DIRPDM : Distributive rational polynomial derivation main variable.
DIRPDR : Distributive rational polynomial derivation.
DIRPHD : Distributive rational polynomial higher derivation.
EVDER : Exponent vector derivation.

derivative

AFPDMV : Algebraic number field polynomial derivative, main variable.
IPDER : Integral polynomial derivative.
IPDMV : Integral polynomial derivative, main variable.
IPFSD : Integral polynomial factorization, second derivative.
IPHDMV : Integral polynomial higher derivative, main variable.
IPSFSD : Integral squarefree factorization, second derivative.
MUPDER : Modular univariate polynomial derivative.
RPDMV : Rational polynomial derivative, main variable.

descriptor

ADDDFDIL : arbitrary domain domain descriptor from distributive polynomial list.
ADDDFDILD : arbitrary domain domain descriptor from distributive polynomial list
ADDDFDIP : arbitrary domain domain descriptor from distributive polynomial.
ADDDFDIPD : arbitrary domain domain descriptor from distributive polynomial or default.
ADDDFSTR : arbitrary domain domain descriptor from string.
ADDDREAD : Arbitrary domain, domain descriptor read.
ADDDWRIT : Arbitrary domain, domain descriptor write.
ADRMDD : arbitrary domain remove domain descriptor informations.
ADVLDD : variable list from domain descriptor.
CdpVd : Case distinction and polynomial set variable list and domain descriptor
CgbVd : Comprehensive Groebner basis variable list and domain descriptor.
DECOFTAG : Descriptor of tagged object.
DVREAD : Polynom descriptor read.
GSDREAD : G-symmetric descriptor read.
GSRDREAD : G-symmetric rational descriptor read.
GsVd : Groebner system variable list and domain descriptor.
INTDDCMP : integer domain descriptor composition.
IPDDADV : integral polynomial domain descriptor advance.
IPDDCMP : integral polynomial domain descriptor composition.
PdVd : Parametric dimension variable list and domain descriptor part.
RFDDADV : rational function domain descriptor advance.
RFDDFIPDD : rational function domain descriptor from integral polynomial domain
SetDdrdFunc : Set domain descriptor read function in domain.
SetDdwrFunc : Set domain descriptor write function in domain.
SetVlddFunc : Set variable list from domain descriptor function in domain.
VdCons : Variable list and domain descriptor construct.
VdD : Variable list and domain descriptor domain descriptor part.
VdD : Variable list and domain descriptor domain descriptor part.
VdParts : Variable list and domain descriptor parts.
VdRead : Variable list and domain descriptor read.
VdV : Variable list and domain descriptor variable list part.

determinant

EXIDET : Exterior integral matrix determinant.
EXIDT2 : Exterior integral matrix determinant 2.
IMDET : Integer matrix determinant, using Gaussian elimination.
IMDETL : Integer matrix determinant, using Laplace expansion.
MAIPDE : Matrix of integral polynomials determinant, exact division algorithm.
MAIPDM : Matrix of integral polynomials determinant, modular algorithm.
MMDDET : Matrix of modular digits determinant.
MMPDMA : Matrix of modular polynomials determinant, modular algorithm.
RNMDET : Rational number matrix determinant, using Gaussian elimination.
RNMDETL : Rational number matrix determinant, using Laplace expansion.

determine

DET : Determine list of polynomials.
DETPOL : Determine polynomial.

difference

ADDIF : Arbitrary domain difference.
AFDIF : Algebraic number field element difference.
AFPDIF : Algebraic number field polynomial difference.
APDIFF : Arbitrary precision floating point difference.
CDIF : Complex number difference.
CLF2 : UGB compute linear form from difference set 2.
CLF3 : UGB compute linear form from difference set 3.
COLDIF : Colour difference.
COMPLF : UGB compute linear form from difference set.
DFP : UGB distributive rational polynomial difference.
DIFF : UGB difference set for rational exponent vector list.
DIFF1 : UGB difference set for two rational exponent vector list.
DIIPDF : Distributive integral polynomial difference.
DIPDIF : Distributive polynomial difference.
DIRPDF : Distributive rational polynomial difference.
EVDFSI : Exponent vector difference and sign.
EVDIF : Exponent vector difference.
EVDIF2 : Exponent vector difference.
FFDIF : Finite field difference.
GBDIFF : Parametric difference.
IDIF : Integer difference.
IMDIF : Integer matrix difference.
IPDIF : Integral polynomial difference.
IVVDIF : Integer vector difference.
MDDIF : Modular digit difference.
MIDIF : Modular integer difference.
MIPDIF : Modular integral polynomial difference.
MKSET : UGB rational exponent vector list difference list.
MPDIF : Modular polynomial difference.
NEWDIF : UGB exponent vector list difference from polynomials.
ODIF : Octonion number difference.
PDIF : UGB rational exponent vector list difference list, incremental.
QDIF : Quaternion number difference.
RFDIF : Rational function difference.

RNDIF : Rational number difference.
RNMDIF : Rational number matrix difference.
RNVDIF : Rational number vector difference.
RNVDIF : UGB rational exponent vector difference.
RPDIF : Rational polynomial difference.
SDIFF : Set difference.
SetDifFunc : Set difference function in domain.
USDIFF : Unordered set difference.
VIDIF : Vector of integers difference.

digit

DAND : Digit and.
DEGCD : Digit extended greatest common divisor.
DGCD : Digit greatest common divisor.
DIGIT : Digit.
DLOG2 : Digit logarithm, base 2.
DNIMP : Digit non-implication.
DNOT : Digit not.
DomLoadMD : Domain load modular digit.
DOMMD : MAS Domain Modular Digit Definition Module.
DOR : Digit or.
DPCC : Digit partial cosequence calculation.
DPGEN : Digit prime generator.
DPR : Digit product.
DQR : Digit quotient and remainder.
DRAN : Digit, random.
DRANN : Digit, random non-negative.
DSQRTF : Digit square root function.
GDPGEN : Gaussian digit prime generator.
IDIPR2 : Integer digit inner product, length 2.
ITD : Integer trailing digit.
MDCRA : Modular digit chinese remainder algorithm.
MDDIF : Modular digit difference.
MDEXP : Modular digit exponentiation.
MDHOM : Modular digit homomorphism.
MDINV : Modular digit inverse.
MDLCRA : Modular digit list chinese remainder algorithm.
MDNEG : Modular digit negative.
MDPROD : Modular digit product.
MDQ : Modular digit quotient.
MDRAN : Modular digit, random.
MDSUM : Modular digit sum.
MIDCRA : Modular integer digit chinese remainder algorithm.
MPMDP : Modular polynomial modular digit product.
SACD : SAC Digit Definition Module.
SACM : SAC Modular Digit and Integer Definition Module.

digits

APNELD : Arbitrary precision floating point number of equal leading digits.
MMDDET : Matrix of modular digits determinant.
MMDNSB : Matrix of modular digits null space basis.

dimension

DIDIMS : Distributive polynomial dimension maximal independent set.
DIDIMWR : Distributive polynomial dimension write.
DILDIM : Distributive polynomial list dimension.
DIMEXE : Parametric dimension exec.
DIMIS : Dimension and maximal independent set.
DIPDIM : DIP Dimension Definition Module.
FdCons : Formula and dimension construct.
FdD : Formula and dimension formula part.
FdF : Formula and dimension formula part.
FdParts : Formula and dimension parts.
FdV : Formula and dimension variable list part.
FdWrite : Formula and dimension write.
GSYSDIM : Groebner system dimension.
MDIM : Matrix dimension.
PdCons : Parametric dimension construct.
PdF : Parametric dimension formula and dimension list part.
PdF : Parametric dimension formula and dimension list part.
PdParts : Parametric dimension parts.
PdVd : Parametric dimension variable list and domain descriptor part.
PdWrite : Parametric dimension write.
PROJ : UGB projection, one dimension.
PROJEC : UGB projection to dimension 1.
WRDIMS : Write dimension.

dimensional

DIPZ : DIP Zero Dimensional Ideal Definition Module.
GBZSET : Groebner base real zero set of zero dimensional ideal.

diophantine

LDSMKB : Linear diophantine system solution, modified Kannan and Bachem algorithm.
LDSSBR : Linear diophantine system solution, based on Rosser ideas.
SACLDIO : SAC Linear Diophantine Equation System Definition Module.

DIP

DIGFET : DIP G base successful extension test.
DIGISM : DIP G base index search for extension multiple univariats.
DIGISR : DIP G base index search for extension reductas.
DIILFRCD : DIP integral list from DIP rational list using common denominator.
DIILFRCD : DIP integral list from DIP rational list using common denominator.
DINNGB : DIP Groebner bases for non noetherian polynomial rings.
DINTFE : DIP normalized tupel field extension.
DINTSR : DIP normalized tupel separation refinement.
DINTSS : DIP normalized tupel strong separation.
DINTZS : DIP nomalized tupels from system zero.
DIPADOM : DIP Arbitrary Domain Definition Module.
DIPAGB : DIP Arbitrary Domain Groebner Basis Definition Module.
DIPC : DIP Common Polynomial System Definition Module.
DIPCJ : DIP Common Polynomial System Definition Module in the sense of Janet.

DIPDCGB : DIP Decompositional Groebner Bases Definition Module.
DIPDCIB : DIP Decompositional Involutive Bases Definition Module.
DIPDDGB : DIP Domain D-Groebner Bases Definition Module.
DIPDEC0 : DIP Ideal Decomposition 0 System Definition Module.
DIPDIM : DIP Dimension Definition Module.
DIPE : DIP Exterior Algebra Definition Module.
DIPGB : DIP Groebner Bases Definition Module.
DIPGCD : DIP GCD Definition Module.
DIPI : DIP Integral Definition Module.
DIPIB : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPIDEAL : DIP Ideal System Definition Module.
DIPIDGB : DIP Integral D-Groebner Bases Definition Module.
DIPIGB : DIP Integral Groebner Bases Definition Module.
DIPIIB : DIP Integral Polynomial System Definition Module in the sense of Janet.
DIPIPOL : DIP Integer Polynomial Definition Module.
DIPRF : DIP Rational Function Definition Module.
DIPRN : DIP Rational Definition Module.
DIPRNGB : DIP Rational Groebner Bases Definition Module.
DIPRNIB : DIP Rational Numbers Polynomial Definition Module in the sense of Janet.
DIPRNPOL : DIP Rational Number Polynomial Definition Module.
DIPROOT : DIP Ideal Real Root System Definition Module.
DIPSP : DIP Integral Function Groebner Bases Implementation Module.
DIPTODEF : DIP define distributive polynomial term order.
DIPTOO : DIP Termorder Optimization Definition Module.
DIPVDEF : DIP define distributive polynomial variable list.
DIPZ : DIP Zero Dimensional Ideal Definition Module.
DIRLPD : DIP rational polynomial ideal primary ideal decomposition.
DIRLPW : DIP rational polynomial ideal primary ideal decomposition write.
DIRPDA : DIP rational polynomial ideal primary ideal decomposition over Q(alpha).
DITFZS : DIP tupel from zero set.
DITSPL : DIP zero set tupel split.

dip

INTDIM : See intdim of dip.

DIP

MASYMDIP : MAS Symbol to DIP Definition Module.
TIPRNGB : DIP Rational Extended Groebner Bases Definition Module.

DIPAGB

SetDIPAGBStrategy : Set the DIPAGB strategy option for the extended critical pair selection.
SetDIPAGBTraceFlag : Set the DIPAGB trace flag.
SetDIPAGBVariableWeight : Set the DIPAGB variable weight list for the normal with sugar strategy.
SetUpdateVariableWeight : Set the DIPAGB variable weight update procedure.
UpdateVariableWeight : Update of the DIPAGB variable weight list.
WriteDIPAGBStrategy : Write the DIPAGB strategy option for the extended critical pair selection
WriteDIPAGBTraceFlag : Write the DIPAGB trace flag in the output stream.
WriteDIPAGBVariableWeight : Write the DIPAGB variable weight list in the output stream.

discrete

DGBRED : Discrete Groebner Base Reduction.
NLDGBRED : Non-Commutative Discrete Groebner Base Reduction.

discriminant

IPDSCR : Integral polynomial discriminant.

disjoint

FORMKVD : formula make variable names disjoint.
MLDMKVD : maslog demonstration make variables disjoint.
PQMKVD : polynomial equation make variable names disjoint.

disjunctive

MLDMKCNF : maslog demonstration make disjunctive normal form.
MLDMKDNF : maslog demonstration make disjunctive normal form.
PQMKCNF : polynomial equation make disjunctive normal form.
PQMKDNF : polynomial equation make disjunctive normal form.

display

DIBUFF : Display input buffer.

distinct

MUPDDF : Modular univariate polynomial distinct degree factorization.
SDR : System of distinct representatives.

distinction

CDINIT : Case distinction init.
CdpCd : Case distinction and polynomial set case distinction part.
CdpCd : Case distinction and polynomial set case distinction part.
CdpCons : Case distinction and polynomial set construct.
CdpParts : Case distinction and polynomial set parts.
CdpPs : Case distinction and polynomial set polynomial set part.
CdpRead : Case distinction and polynomial set read.
CdpVd : Case distinction and polynomial set variable list and domain descriptor
CdpWrite : Case distinction and polynomial set write.
CdRead : Case distinction read.
CdWrite : Case distinction write.
CgbCd : Groebner system initial case distinction.
CONINI : Initialize case distinction.
DWRIT : Distinction write.
GsCd : Groebner system initial case distinction.
UPDCAS : Update case distinction.

distributiv

DIPCLP : Distributiv Polynomial Class of Polynomial.
DIPCLT : Distributiv Polynomial Class of Term.
DIRPMV : Distributiv Polynomial multiplication with a variable.

distributive

AD2DIP	: arbitrary domain to distributive polynomial.
ADDDFDIL	: arbitrary domain domain descriptor from distributive polynomial list.
ADDDFDILD	: arbitrary domain domain descriptor from distributive polynomial list
ADDDFDIP	: arbitrary domain domain descriptor from distributive polynomial.
ADDDFDIPD	: arbitrary domain domain descriptor from distributive polynomial or default.
ADDNFDIL	: arbitrary domain domain number from distributive polynomial list.
ADDNFDILD	: arbitrary domain domain number from distributive polynomial list
ADDNFDIP	: arbitrary domain domain number from distributive polynomial.
ADDNFDIPD	: arbitrary domain domain number from distributive polynomial or default.
ADDNFEDIP	: Arbitrary domain domain number from extended distributive polynomial.
ADPFDIP	: arbitrary domain polynomial from distributive polynomial.
CGBLM	: CGB coloured distributive polynomial list merge.
DFP	: UGB distributive rational polynomial difference.
DIDIMS	: Distributive polynomial dimension maximal independent set.
DIDIMWR	: Distributive polynomial dimension write.
DIDPALCMPC	: Distributive domain polynomial array list check and mark polynomials.
DIDPCPLMS1	: Distributive domain polynomial list construct pairs list merge sort.
DIDPDGB	: Distributive domain polynomial D-groebner basis.
DIDPDNF	: Distributive domain polynomial D-normal form.
DIDPEGB	: Distributive domain polynomial E-groebner basis.
DIDPELIMDGB	: Distributive domain polynomial eliminate D-groebner base.
DIDPENF	: Distributive domain polynomial E-normal form.
DIDPGPOL	: Distributive domain polynomial g polynomial.
DIDPLCPL4	: Distributive domain polynomial list construct pair list.
DIDPLEXTAL	: Distributive domain polynomial list extend array list.
DIDPLM1	: Distributive domain polynomial list merge sort.
DIDPREDDGB	: Distributive domain polynomial reduce D-groebner base.
DIDPSPOL	: Distributive domain polynomial s polynomial.
DIDPSPOL2	: Distributive domain polynomial s polynomial.
DIDPTDR	: Distributive domain polynomial top-D-reduzibel.
DIDPUCPL1	: Distributive domain polynomial update constructed pairs list.
DIFIP	: Distributive polynomial from distributive integral polynomial.
DIFIP	: Distributive polynomial from distributive integral polynomial.
DIFPF	: Distributive polynomial with arbitrary domain coefficients from
DIGBC3	: Distributive polynomial groebner basis criterion 3.
DIGBC4	: Distributive polynomial groebner basis criterion 4.
DIGBMI	: Distributive minimal ordered groebner basis.
DIGBSI	: Distributive polynomial system algebraic number G basis sign.
DIGBZT	: Distributive polynomial groebner base common zero test.
DIGMIN	: Distributive minimal ordered groebner basis.
DIIFGB	: Distributive integral function polynomial groebner basis.
DIIFLS	: Distributive integral function polynomial list irreducible set.
DIIFMI	: Distributive minimal ordered groebner basis.
DIIFNF	: Distributive integral function polynomial normal form.
DIIFRP	: Distributive integral polynomial from rational polynomial.
DIIFSP	: Distributive integral function polynom S-polynomial.
DIIGBA	: Distributive integral polynomial groebner basis augmentation.
DIIGMI	: Distributive minimal ordered groebner basis.

DIILFR : Distributive integral polynomial list from rational polynomial list.
DIILIS : Distributive integral polynomial list irreducible set.
DIILIS : Distributive integral polynomial list irreducible set.
DIILISJ : Distributive integral polynomial list irreducible set.
DIILPP : Distributive integral polynomial list primitive part.
DIILRD : Distributive integral polynomial list read.
DIILWR : Distributive integral polynomial list write.
DIIPAB : Distributive integral polynomial absolute value.
DIIPALCMPC : Distributive integral polynomial array list check and mark polynomials.
DIIPCOM : Distributive integral polynomial complete system.
DIIPCP : Distributive integral polynomial content and primitive part.
DIIPCPLMS1 : Distributive integral polynomial list construct pairs list merge sort.
DIIPDF : Distributive integral polynomial difference.
DIIPDGB : Distributive integral polynomial D-groebner basis.
DIIPDM : Distributive integral polynomial derivation main variable.
DIIPDNF : Distributive integral polynomial normal form.
DIIPDR : Distributive integral polynomial derivation.
DIIPEGB : Distributive integral polynomial E-groebner basis.
DIIPELIMDGB : Distributive integral polynomial eliminate D-groebner base.
DIIPEM : Distributive integral polynomial evaluation of main variable.
DIIPENF : Distributive integral polynomial e-normal form.
DIIPEV : Distributive integral polynomial evaluation of the i-th variable.
DIIPEX : Distributive integral polynomial exponentiation.
DIIPGB : Distributive integral polynomial groebner basis.
DIIPGPOL : Distributive integral polynomial g polynomial.
DIIPHD : Distributive integral polynomial higher derivation.
DIIPIB : Distributive integral involutive base.
DIIPIB2 : Distributive integral polynomial involutive base.
DIIPIB3 : Distributive integral polynomial involutive base.
DIIPIP : Distributive integral polynomial integer product.
DIIPIQ : Distributive integral polynomial integer quotient.
DIIPLCPL4 : Distributive integral polynomial list construct pair list.
DIIPLEXTAL : Distributive integral polynomial list extend array list.
DIIPLM1 : Distributive integral polynomial list merge sort.
DIIPLS : Distributive integral polynomial list sum.
DIIPMN : Distributive integral polynomial maximum norm.
DIIPNF : Distributive integral polynomial normal form.
DIIPNFJ : Distributive integral polynomial normal form in the sense of Janet.
DIIPNG : Distributive integral polynomial negative.
DIIPNORM : Distributive integral polynomial norm.
DIIPON : Distributive integral polynomial one.
DIIPPR : Distributive integral polynomial product.
DIIPPR2 : Distributive integral polynomial product.
DIIPPS : Distributive integral polynomial pseudo-remainder.
DIIPQ : Distributive integral polynomial quotient.
DIIPQR : Distributive integral polynomial quotient and remainder.
DIIPRA : Distributive integral polynomial random.
DIIPRD : Distributive integral polynomial read.
DIIPREDDGB : Distributive integral polynomial reduce D-groebner base.
DIIPSG : Distributive integral polynomial sign.
DIIPSM : Distributive integral polynomial sum.
DIIPSN : Distributive integral polynomial sum norm.

DIIPSO : Distributive integral polynomial sort.
DIIPSP : Distributive integral polynomial s polynomial.
DIIPSPOL : Distributive integral polynomial s polynomial.
DIIPSPOL2 : Distributive integral polynomial s polynomial.
DIIPSU : Distributive integral polynomial substitution.
DIIPSV : Distributive integral polynomial substitution for main variable.
DIIPTDR : Distributive integral polynomial top-D-reduzibel.
DIIPTM : Distributive integral polynomial translation main variable.
DIIPTR : Distributive integral polynomial translation.
DIIPUCPL1 : Distributive polynomial D-update constructed pairs list.
DIIPWR : Distributive integral polynomial write.
DIIPWV : Distributive integral polynomial write with standard variable list.
DIIRAS : Distributive integral polynomial random sparse exponent vector.
DIITNT : Distributive polynomial system interval tupel from norm tupel.
DIITWR : Distributive polynomial system interval tupels write.
DILADNF : distributive polynomial list arbitrary domain normal form.
DILATDG : Distributive polynom list add total degree.
DILBBS : Distributive List Bubble Sort.
DILBSO : Distributive polynomial list bubble sort.
DILCAN : Distributive Polynomial Cancel.
DILCONV : distributive polynomial list conversion.
DILCPL : Distributive polynomial list construct pair list.
DILDIM : Distributive polynomial list dimension.
DILEBBS : Distributive List Extended Bubble Sort.
DILEP2P : Distributive polynom list extended polynom to polynom.
DILFDILP : distributive polynomial list from distributive polynomial list over
DILFDILP : distributive polynomial list from distributive polynomial list over
DILFEL : Distributive polynomial list from exponent vector list.
DILFPFL : Distributive polynomial list with arbitrary domain coefficients from
DILFPL : Distributive polynomial list from polynom list.
DILIMO : distributive polynomial list inverse monomial order.
DILINV : distributive polynomial list introduce new variable.
DILIS : Distributive polynomial list irreducible set.
DILISJ : Distributive polynomial list irreducible set in the sense of Janet.
DILISJ2 : Distributive polynomial list irreducible set.
DILMOC : distributive polynomial monic.
DILNFJ : Distributive Polynomial List normalform in the sense of Janet.
DILPERM : distributive polynomial list permutation of variables.
DILPFDIL : distributive polynomials over polynomial ring list from distributive
DILPFDIL : distributive polynomials over polynomial ring list from distributive
DILPROD : distributive polynomial list product.
DILRD : Distributive polynomial list read.
DILSUM : Distributive polynomial list sum.
DILTDG : Distributive polynomial list total degree
DILUPL : Distributive polynomial list update pair list.
DILWR : Distributive polynomial list write.
DIMPAD : distributive monomial partial derivation.
DIN1GB : Distributive non-commutative polynomials Groebner base.
DINCCO : Distributive rational non-commutative polynomial commutator.
DINCCP : Distributive rational non-commutative polynomial center polynomial.
DINCCPpre : Distributive rational non-commutative polynomial center polynomial preparation.
DINCGB : Distributive non-commutative polynomials Groebner base.

DINLGB : Distributive non-commutative polynomials left Groebner base.
DINLGM : Distributive non-commutative minimal ordered left Groebner base.
DINLIS : Distributive non-commutative polynomial list left irreducible set.
DINLMPG : Distributive non-commutative left rational minimal polynomial for a G basis.
DINLMPL : Distributive non-commutative left rational minimal polynomial list for a G basis.
DINLNF : Distributive non-commutative polynomial left normal form.
DINLRD : Distributive non-commutative polynomial list read.
DINLSP : Distributive non-commutative polynomial left S-polynomial.
DINPEX : Distributive non-commutative polynomial exponentiation.
DINPPR : Distributive polynomial non-commutative product.
DINPQ : Distributive non-commutative Polynomial Quotient.
DINPRD : Distributive non-commutative polynomial read.
DINPTL : Distributive polynomial non-commutative product table lookup.
DINPTslT : Distributive polynomial non-commutative product table strict lex test.
DINPTU : Distributive polynomial non-commutative product table update.
DINTWR : Distributive polynomial system normalized tupels write.
DIP2AD : distributive polynomial to arbitrary domain.
DIP2SYM : Distributive polynomial to symbol term.
DIPADGB : distributive polynomial arbitrary domain groebner basis.
DIPADGBext : distributive polynomial arbitrary domain groebner basis extension.
DIPADGBRED : distributive polynomial groebner basis reduction.
DIPADGBunion : distributive polynomial arbitrary domain groebner basis union.
DIPADIRSET : distributive polynomial arbitrary domain irreducible set.
DIPADM : Distributive polynomial advance main variable.
DIPADNF : distributive polynomial arbitrary domain normal form.
DIPADS : Distributive polynomial advance and substitute.
DIPADV : Distributive polynomial advance.
DIPAGB : Distributive polynomial arbitrary domain Groebner basis.
DIPBCP : Distributive polynomial base coefficient product.
DIPBSO : Distributive polynomial bubble sort.
DIPCMP : Distributive polynomial composition.
DIPCNST : distributive polynomial is constant.
DIPCNSTR : distributive polynomial constant relative to variables.
DIPCOM : Distributive polynomial complete.
DIPCONV : distributive polynomial conversion.
DIPCPP : distributive polynomial content and primitive part.
DIPCT : distributive polynomial coefficient tuple.
DIPDEG : Distributive polynomial degree.
DIPDEGI : distributive polynomial degree of i-th main variable.
DIPDEM : Distributive polynomial degree matrix.
DIPDEV : Distributive polynomial degree vector.
DIPDIF : Distributive polynomial difference.
DIPDPV : Distributive polynomial division by power of variable.
DIPEINFJ : Integral distributive extended polynomial normal form in the sense of Janet.
DIPENFJ : Distributive extended polynomial normal form in the sense of Janet.
DIPERM : Distributive polynomial permutation of variables.
DIPEVL : Distributive polynomial exponent vector leading monomial.
DIPEVP : Distributive polynomial exponent vector product.
DIPEXC : Distributive polynomial exchange variables.
DIPEXP : Distributive polynomial exponentiation.

DIPEXTEND : Distributive polynomial extension.
DIPFAC : distributive polynomial factorization.
DIPFADIP : distributive polynomial from arbitrary domain integral polynomial.
DIPFDIPP : distributive polynomial from distributive polynomial over polynomial ring.
DIPFDIPP : distributive polynomial from distributive polynomial over polynomial ring.
DIPFIP : distributive polynomial from integral polynomial.
DIPFIRST : Distributive polynomial first polynomial,
DIPFMO : Distributive polynomial from monomial.
DIPFP : Distributive polynomial from polynomial.
DIPGB : Distributive polynomial groebner basis.
DIPIB : Distributive polynomial involutive basis.
DIPIB2 : Distributive polynomial involutive basis.
DIPIB3 : Distributive polynom involutive base.
DIPIB4 : Distributive polynomial involutive basis.
DIPIMO : distributive polynomial inverse monomial order.
DIPINFJ : Integral Distributive polynomial normal form in the sense of Janet.
DIPINFJS : Integral Distributive polynomial normal form in the sense of Janet modulo a
DIPINV : Distributive polynomial introduction of new variables.
DIPIRL : distributive polynomials interreduced list of polynomials.
DIPIRLJ : distributive polynomial interreduced list in the sense of Janet.
DIPIRLJ2 : Distributive polynomial list interreduced list in the sense of Janet.
DIPLBC : Distributive polynomial leading base coefficient.
DIPLDC : Distributive polynomial leading coefficient.
DIPLDM : Distributive polynomial list degree matrix.
DIPLDV : Distributive polynomial list dependency on variables.
DIPLFPFL : Distributive polynomial list from recursive polynomial
DIPLIR : distributive polynomial list interreduce.
DIPLM : Distributive polynomial list merge.
DIPLMD : distributive polynomial list maximum degree.
DIPLPM : Distributive polynomial list pair-merge sort.
DIPLRS : Distributive polynomial list re-sort.
DIPMAD : Distributive polynomial monomial advance.
DIPMC2 : UGB distributive polynomial composition 2.
DIPMCP : Distributive polynomial monomial composition.
DIPMOC : Distributive polynomial monic.
DIPMPM : Distributive polynomial multiplication by power of main variable.
DIPMPV : Distributive polynomial multiplication by power of variable.
DIPMRD : Distributive polynomial monomial reductum.
DIPMST : Distributive polynomial monomial set.
DIPMVV : distributive polynomial minimal variable vector.
DIPNBC : Distributive polynomial number of base coefficients.
DIPNEG : Distributive polynomial negative.
DIPNF : distributive polynomial normalform.
DIPNFJ : Distributive polynomial normal form in the sense of Janet.
DIPNFJS : Distributive polynomial normal form in the sense of Janet modulo a set of
DIPNML : Distributive polynomial nonmultiple variable list.
DIPNOR : Distributive polynomial normal form.
DIPNOV : Distributive polynomial number of variables.
DIPONE : distributive polynomial arbitrary domain one.

DIPPAD : distributive polynomial partial derivation.
DIPPCPP : distributive polynomial pseudo content and primitive part.
DIPPFDIP : distributive polynomial over polynomial ring from distributive polynomial.
DIPPFDIP : distributive polynomial over polynomial ring from distributive polynomial.
DIPPGL : Distributive polynomial prolongation list.
DIPPGL2 : Distributive polynomial prolongation list.
DIPPGL3 : Distributive polynom prolongation list.
DIPPOWER : distributive polynomial power.
DIPQR : Distributive polynomial quotient and remainder.
DIPRED : Distributive polynomial reductum.
DIPRLF : distributive polynomials reduce list of polynomials with factor.
DIPROD : Distributive polynomial product.
DIPRWTDG : Distributive polynomial rational-weighted total degree.
DIPS : distributive polynomial S-polynomial.
DIPSFF : distributive polynomial squarefree factorization.
DIPSP : Distributive polynomial S-polynomial.
DIPSSM : Distributive polynomial sort and select minimal.
DIPSUM : Distributive polynomial sum.
DIPTBC : Distributive polynomial trailing base coefficient.
DIPTCF : Distributive polynomial trailing coefficient.
DIPTCS : Distributive polynomial trailing coefficient specified variable.
DIPTDG : Distributive polynomial total degree.
DIPTODEF : DIP define distributive polynomial term order.
DIPTOOLS : Distributive Polynomials Tools Definition Module.
DIPTRM : Distributive polynomial terms.
DIPTYP : Distributive polynomial typ.
DIPUNT : Distributive polynomial univariate test.
DIPUV : Distributive polynomial univariate variable output.
DIPVDEF : DIP define distributive polynomial variable list.
DIPVL : Distributive Polynomial List of Variables.
DIPVOP : Distributive polynomial variable ordering optimisation.
DIPVOPP : Distributive polynomial variable ordering optimization and permutation
DIPXCM : distributive polynomial extract constant monomials.
DIREAD : Distributive polynomial read.
DIREGB : Distributive rational polynomials extended groebner basis.
DIRFAC : Distributive rational polynomial factorisation.
DIRFIP : Distributive rational polynomial from integral polynomial.
DIRGBA : Distributive rational polynomial groebner basis augmentation.
DIRGBR : Distributive rational polynomial groebner basis recursion.
DIRGZS : Distributive rational Groebner base zero set.
DIRLCT : Distributive rational polynomial list ideal containment test.
DIRLIP : Distributive rational polynomial list ideal product.
DIRLIS : Distributive rational polynomial list irreducible set.
DIRLISJ : Distributive rational polynomial list irreducible set.
DIRLPI : Distributive rational polynomial list primary ideal.
DIRLRD : Distributive rational polynomial list read.
DIRLWR : Distributive rational polynomial list write.
DIRMPG : Distributive rational minimal polynomial for a groebner basis.
DIROWR : Distributive polynomial system real root write.
DIRPAB : Distributive rational polynomial absolute value.

DIRPCOM : Distributive rational polynom complete system.
DIRPDF : Distributive rational polynomial difference.
DIRPDM : Distributive rational polynomial derivation main variable.
DIRPDR : Distributive rational polynomial derivation.
DIRPEM : Distributive rational polynomial evaluation of main variable.
DIRPES : Distributive rational polynomial elementary symmetric functions.
DIRPEV : Distributive rational polynomial evaluation of the i-th variable.
DIRPEX : Distributive rational polynomial exponentiation.
DIRPFT : Distributive rational polynomial from term.
DIRPGB : Distributive rational polynomials groebner basis.
DIRPHD : Distributive rational polynomial higher derivation.
DIRPIB2 : Distributive rational polynom involutive basis.
DIRPLS : Distributive rational polynomial list sum.
DIRPMC : Distributive rational polynomial monic.
DIRPMN : Distributive rational polynomial maximum norm.
DIRPNF : Distributive rational polynomial normal form.
DIRPNFJ : Distributive rational polynomial normal form in the sense of Janet.
DIRPNG : Distributive rational polynomial negative.
DIRPON : Distributive rational polynomial one.
DIRPPR : Distributive rational polynomial product.
DIRPQ : Distributive rational polynomial quotient.
DIRPQR : Distributive rational polynomial quotient and remainder.
DIRPRA : Distributive rational polynomial random.
DIRPRD : Distributive rational polynomial read.
DIRPRP : Distributive rational polynomial rational number product.
DIRPRQ : Distributive rational polynomial rational number quotient.
DIRPSE : Distributive rational polynomial symm.
DIRPSG : Distributive rational polynomial sign.
DIRPSM : Distributive rational polynomial sum.
DIRPSN : Distributive rational polynomial sum norm.
DIRPSO : Distributive rational polynomial sort.
DIRPSP : Distributive rational polynomial S polynomial.
DIRPSP : UGB distributive polynomial S-polynomial.
DIRPSR : Distributive rational polynomial symmetric function reduction.
DIRPSU : Distributive rational polynomial substitution.
DIRPSV : Distributive rational polynomial substitution for main variable.
DIRPTM : Distributive rational polynomial translation main variable.
DIRPTR : Distributive rational polynomial translation.
DIRPWR : Distributive rational polynomial write.
DIRPWV : Distributive rational polynomial write with standard variable
DIRRAS : Distributive rational polynomial, random sparse exponent vector.
DIRRNF : UGB distributive polynomial normalform.
DIWRIT : Distributive polynomial write.
DNLCPL : distributive polynomial non-noetherian left construct pair list.
DNLUPL : distributive polynomial non-noetherian left update pair list.
DNN2GB : distributive polynomials non-noetherian 2-sided Groebner base.
DNNLGB : distributive non-noetherian polynomials left Groebner base.
DNNRGB : distributive polynomials non-noetherian right Groebner base.
DNRCPL : distributive polynomial non-noetherian right construct pair list.
DNRUPL : distributive polynomial non-noetherian right update pair list.
ECPPOLY1 : Extended critical pair select the first extended distributive polynomial.
ECPPOLY2 : Extended critical pair select the second extended distributive

EDIIFSUGNF : Extended distributive integral function polynomial normal with sugar
EDIIFSUGSP : Extended distributive integral function polynomial normal with sugar
EDIPEVL : Extended distributive polynomial exponent vector of the leading monomial.
EDIPNOR : Extended distributive polynomial normal form.
EDIPSP : Extended distributive polynomial S-polynomial.
EDIPSUGAR : Extended distributive polynomial sugar.
EDIPSUGCON : Extended distributive polynomial normal with sugar strategy constructor.
EDIPSUGNOR : Extended distributive polynomial normal with sugar strategy normal form.
EDIPSUGSP : Extended distributive polynomial normal with sugar strategy S-polynomial.
EDIPUNEXTEND : Extended distributive polynomial un-extend.
EDIPWRITE : Extended distributive polynomial write.
GroebnerBases1 : Distributive polynomials decompositional groebner bases 1.
GroebnerBases2 : Distributive polynomials decompositional groebner bases 2.
HDIFDI : Homogeneous distributive polynomial from distributive polynomial.
HDIFDI : Homogeneous distributive polynomial from distributive polynomial.
InitDIPIIB : Init distributive integral involutive base.
LDEG : Distributive polynomial list total degree.
LDIPEXTEND : List of distributive polynomials extend.
LEDIPUNEXTEND : List of extended distributive polynomials un-extend.
LEDIPWRITE : List of extended distributive polynomials write.
PCOMP : UGB distributive polynomial composition.
PFDIP : Polynomial from distributive polynomial.
PFLDIPL : Recursive polynomial list (with domain-descriptor) from distributive
PLFDIL : Polynomial list from distributive polynom list.
SetDIPExtend : Set the distributive polynomial extension function.
SetDIPIBCancel : Set distributive polynomial involutive base cancel.
SetDIPIBCrit : Set distributive polynomial involutive base critera.
SetDIPIBISJ : Set distributive involutive base irreducible set in the sense of Janet.
SetDIPIBopt : Set distributive polynomial involutive base options.
SetDIPIBSelect : Set distributive polynomial involutive base select.
SetDIPIIBSelect : Set Distributive integral polynomial Select.
SetEDIPNormalform : Set the extended distributive polynomial normalform function.
SetEDIPSPolynomial : Set the extended distributive S-polynomial function.
SetEDIPUnExtend : Set the extended distributive polynomial un-extension function.
SetEDIPWrite : Set the extended distributive polynomial write procedure.
SFP : UGB distributive rational polynomial sum.
SYM2DIP : Symbol term to distributive polynomial.
TFDIRP : Term from distributive rational polynomial.
XDIPFPF : Distributive polynomial from recursive polynomial (with domain-descriptor).
XPFDIP : Recursive polynomial (with domain-descriptor) from distributive polynomial.

divided

PDBORD : Polynomial divided by order.

division

DIPDPV : Distributive polynomial division by power of variable.
IDP2 : Integer division by power of 2.
MAIPDE : Matrix of integral polynomials determinant, exact division algorithm.
PDPV : Polynomial division by power of variable.

divisor

ADGCD : Arbitrary domain greatest common divisor.
ADGCDC : Arbitrary domain greatest common divisor and cofactors.
ADGCDE : Arbitrary domain greatest common divisor and linear combination.
AFUPGC : Algebraic number field univariate polynomial greatest common divisor
AFUPGS : Algebraic number field polynomial greatest squarefree divisor.
DEGCD : Digit extended greatest common divisor.
DGCD : Digit greatest common divisor.
EVGCD : Exponent vector greatest common divisor.
IDEGCD : Integer doubly extended greatest common divisor algorithm.
IEGCD : Integer extended greatest common divisor algorithm.
IGCD : Integer greatest common divisor.
IGCDCF : Integer greatest common divisor and cofactors.
IHEGCD : Integer half-extended greatest common divisor.
ILPDS : Integer large prime divisor search.
IMPDS : Integer medium prime divisor search.
IPGCDC : Integral polynomial greatest common divisor and cofactors.
IPPGSD : Integral polynomial primitive greatest squarefree divisor.
MPGCDC : Modular polynomial greatest common divisor and cofactors.
MUPEGC : Modular univariate polynomial extended greatest common divisor.
MUPGCD : Modular univariate polynomial greatest common divisor.
MUPHEG : Modular univariate polynomial half-extended greatest common divisor.
RPBLGS : Rational polynomial base coefficients least common multiple, greatest common divisor, and sign.
RUPEGC : Rational univariate polynomial extended greatest common divisor.
RUPGCD : Rational univariate polynomial greatest common divisor.
RUPHEG : Rational univariate polynomial half-extended greatest common divisor.

divisors

ISPD : Integer small prime divisors.

dnf

FORMKDNF : formula make dnf.
PQDnfSimplify : polynomial equation dnf based simplification.

do

DoParse : Do parse.
DoWrite : Do Write.

domain

AD2DIP : arbitrary domain to distributive polynomial.
ADABSF : Arbitrary domain absolute value.
ADCAST : arbitraray domain cast.
ADCHARPOL : Arbitrary domain characteristic polynomial.
ADCNST : Arbitrary domain constant test.
ADCOMP : Arbitrary domain comparison.
ADCONV : Arbitrary domain conversion.
ADDDFDIL : arbitrary domain domain descriptor from distributive polynomial list.
ADDDFDIL : arbitrary domain domain descriptor from distributive polynomial list.
ADDDFDILD : arbitrary domain domain descriptor from distributive polynomial list
ADDDFDILD : arbitrary domain domain descriptor from distributive polynomial list
ADDDFDIP : arbitrary domain domain descriptor from distributive polynomial.
ADDDFDIP : arbitrary domain domain descriptor from distributive polynomial.
ADDDFDIPD : arbitrary domain domain descriptor from distributive polynomial or default.
ADDDFDIPD : arbitrary domain domain descriptor from distributive polynomial or default.
ADDDFSTR : arbitrary domain domain descriptor from string.
ADDDFSTR : arbitrary domain domain descriptor from string.
ADDDREAD : Arbitrary domain, domain descriptor read.
ADDDREAD : Arbitrary domain, domain descriptor read.
ADDDWRIT : Arbitrary domain, domain descriptor write.
ADDDWRIT : Arbitrary domain, domain descriptor write.
ADDIF : Arbitrary domain difference.
ADDNFDIL : arbitrary domain domain number from distributive polynomial list.
ADDNFDIL : arbitrary domain domain number from distributive polynomial list.
ADDNFDILD : arbitrary domain domain number from distributive polynomial list
ADDNFDILD : arbitrary domain domain number from distributive polynomial list
ADDNFDIP : arbitrary domain domain number from distributive polynomial.
ADDNFDIP : arbitrary domain domain number from distributive polynomial.
ADDNFDIPD : arbitrary domain domain number from distributive polynomial or default.
ADDNFDIPD : arbitrary domain domain number from distributive polynomial or default.
ADDNFEDIP : Arbitrary domain domain number from extended distributive polynomial.
ADDNFEDIP : Arbitrary domain domain number from extended distributive polynomial.
ADEPCompose : Arbitrary domain extended polynomial compose.
ADEPCrumble : Arbitrary domain extended polynomial crumble.
ADEPdegree : Arbitrary domain extended polynomial degree.
ADEPheadterm : Arbitrary domain extended plynomial head-term.
ADEPmark : Arbitrary domain extended polynomial mark.
ADEPNFJ : Arbitrary domain extended polynomial normalform in the sense of Janet.
ADEPpolynomial : Arbitrary domain extended polynomial polynomial.
ADEPtriple : Arbitrary domain extende polynomial triple.
ADEPuntriple : Arbitrary domain extended polynomial untriple.
ADEXP : Arbitrary domain exponentiation.
ADEXTRA : Arbitrary domain extra definition module.
ADFACT : Arbitrary domain factorization.
ADFACTO : Arbitrary domain factorization with variable order optimization.

ADFI : Arbitrary domain from integer.
ADFIP : Arbitrary domain from integral polynomial.
ADGCD : Arbitrary domain greatest common divisor.
ADGCDC : Arbitrary domain greatest common divisor and cofactors.
ADGCDE : Arbitrary domain greatest common divisor and linear combination.
ADGJredI : Arbitrary domain polynomial G Janet-reducible modulo I.
ADINV : Arbitrary domain inverse.
ADINVT : Arbitrary domain inverse existence test.
ADIredG : Arbitrary domain polynomial set I reducible modulo G.
ADLCM : Arbitrary domain least common multiple.
ADLGeqH : Arbitrary domain polynomial list G equal H.
ADLGinH : Arbitrary domain polynomial list G in H.
AdLoadConvFunc : arbitrary domain load conversion functions.
ADMPROD : Arbitrary domain matrix product.
ADMPTRACE : Arbitrary domain matrix product trace.
ADMSUM : Arbitrary domain matrix sum.
ADMTRACE : Arbitrary domain matrix trace.
ADMWRITE : Arbitrary domain matrix write.
ADNEG : Arbitrary domain negative.
ADONE : Arbitrary domain one.
ADPCP : Arbitrary Domain polynomial content and primitive part.
ADPCPP : Arbitrary domain polynomial content and primitive part.
ADPFACT : Arbitrary domain polynomial factorization.
ADPFDIP : arbitrary domain polynomial from distributive polynomial.
ADPFeqG : Arbitrary domain polynomial F equal G.
ADPIQ : Arbitrary domain polynomial integer quotient.
ADPNEG : Arbitrary domain polynomial negative.
ADPNF : Arbitrary domain polynomial normalform.
ADPNFJ : Arbitrary domain polynomial normalform in the sense of Janet.
ADPNFJS : Arbitrary domain polynomial normalform in the sense of Janet modulo a set
ADPROD : Arbitrary domain product.
ADPSFF : Arbitrary domain polynomial squarefree factorization.
ADPSP : Arbitrary domain polynomial S-polynomial.
ADPSUGNF : Arbitrary domain polynomial normal with sugar strategy normalform.
ADPSUGSP : Arbitrary domain polynomial normal with sugar strategy S-polynomial.
ADQR : Arbitrary domain quotient and remainder.
ADQUOT : Arbitrary domain quotient.
ADREAD : Arbitrary domain read.
ADREM : Arbitrary domain remainder.
ADRFFADIP : arbitrary domain rational function from arbitrary domain intgeral
ADRFFADIP : arbitrary domain rational function from arbitrary domain intgeral
ADRMDD : arbitrary domain remove domain descriptor informations.
ADRMDD : arbitrary domain remove domain descriptor informations.
ADSIG : Arbitrary domain signature.
ADSIGN : Arbitrary domain sign.
ADSMPROD : Arbitrary domain scalar and matrix product.
ADSUM : Arbitrary domain sum.
ADTOIP : Arbitrary domain to integral polynomial conversion.
ADTOOLS : Arbitrary Domain Tools Definition Module.
ADUM : Arbitrary domain unit matrix.
ADVLDD : variable list from domain descriptor.

ADVSPROD : Arbitrary domain vector scalar product.
ADVSVPROD : Arbitrary domain vector scalar vector product.
ADVVSUM : Arbitrary domain vector vector sum.
ADVWRITE : Arbitrary domain vector write.
ADWRIT : Arbitrary domain write.
CdpVd : Case distinction and polynomial set variable list and domain descriptor
CgbVd : Comprehensive Groebner basis variable list and domain descriptor.
CHDOM : Change domain.
DIDPALCMPC : Distributive domain polynomial array list check and mark polynomials.
DIDPCPLMS1 : Distributive domain polynomial list construct pairs list merge sort.
DIDPDGB : Distributive domain polynomial D-groebner basis.
DIDPDNF : Distributive domain polynomial D-normal form.
DIDPEGB : Distributive domain polynomial E-groebner basis.
DIDPELIMDGB : Distributive domain polynomial eliminate D-groebner base.
DIDPENF : Distributive domain polynomial E-normal form.
DIDPGPOL : Distributive domain polynomial g polynomial.
DIDPLCPL4 : Distributive domain polynomial list construct pair list.
DIDPLEXTAL : Distributive domain polynomial list extend array list.
DIDPLM1 : Distributive domain polynomial list merge sort.
DIDPREDDGB : Distributive domain polynomial reduce D-groebner base.
DIDPSPOL : Distributive domain polynomial s polynomial.
DIDPSPOL2 : Distributive domain polynomial s polynomial.
DIDPTDR : Distributive domain polynomial top-D-reduzibel.
DIDPUCPL1 : Distributive domain polynomial update constructed pairs list.
DIFPF : Distributive polynomial with arbitrary domain coefficients from
DILADNF : distributive polynomial list arbitrary domain normal form.
DILFPFL : Distributive polynomial list with arbitrary domain coefficients from
DIP2AD : distributive polynomial to arbitrary domain.
DIPADGB : distributive polynomial arbitrary domain groebner basis.
DIPADGBext : distributive polynomial arbitrary domain groebner basis extension.
DIPADGBunion : distributive polynomial arbitrary domain groebner basis union.
DIPADIRSET : distributive polynomial arbitrary domain irreducible set.
DIPADNF : distributive polynomial arbitrary domain normal form.
DIPADOM : DIP Arbitrary Domain Definition Module.
DIPAGB : DIP Arbitrary Domain Groebner Basis Definition Module.
DIPAGB : Distributive polynomial arbitrary domain Groebner basis.
DIPDDGB : DIP Domain D-Groebner Bases Definition Module.
DIPFADIP : distributive polynomial from arbitrary domain integral polynomial.
DIPONE : distributive polynomial arbitrary domain one.
DOMAF : MAS Domain Algebraic Number Definition Module.
DOMAPF : MAS Domain Arbitrary Precision Floating Point Definition Module.
DOMC : MAS Domain Complex Number Definition Module.
DOMFF : MAS Domain Finite Field Definition Module.
DOMI : MAS Domain Integer Definition Module.
DOMIP : MAS Domain Integral Polynomial Definition Module.
DomLoadAF : Domain load algebraic number.
DomLoadAPF : Domain load arbitrary precision floating point.
DomLoadC : Domain load complex number.
DomLoadFF : Domain load finite field.
DomLoadI : Domain load integer.
DomLoadIP : Domain load integral polynomials .

DomLoadMD : Domain load modular digit.
DomLoadMI : Domain load modular integer.
DomLoadO : Domain load octonion number.
DomLoadQ : Domain load quaternion number.
DomLoadRF : Domain load rational function.
DomLoadRN : Domain load rational number.
DomLoadRP : Domain load rational polynomials .
DOMMD : MAS Domain Modular Digit Definition Module.
DOMMI : MAS Domain Modular Integer Definition Module.
DOMO : MAS Domain Octonion Number Definition Module.
DOMQ : MAS Domain Quaternion Number Definition Module.
DOMRF : MAS Domain Rational Function Definition Module.
DOMRN : MAS Domain Rational Number Definition Module.
DOMRP : MAS Domain Rational Polynomial Definition Module.
DomSummary : Arbitrary domain summary.
GsVd : Groebner system variable list and domain descriptor.
InitExternalsE : Initialize external compiled arbitrary domain procedures.
INTDDCMP : integer domain descriptor composition.
IPDDADV : integral polynomial domain descriptor advance.
IPDDCMP : integral polynomial domain descriptor composition.
IPDECMP : integral polynomial domain element composition.
MASADOM : MAS Arbitrary Domain Definition Module.
NewDom : New domain.
PdVd : Parametric dimension variable list and domain descriptor part.
RFDDADV : rational function domain descriptor advance.
RFDDFIPDD : rational function domain descriptor from integral polynomial domain
RFDDFIPDD : rational function domain descriptor from integral polynomial domain
RRADCOUNT : Real root arbirary domain count.
RRADNFORM : Real root arbitrary domain normal form.
RRADOM : Real Root Arbitrary Domain Definition Module.
RRADPOLMATRIX : Real root arbitrary domain polynomial matrix.
RRADQUADFORM : Real root arbitrary domain quadratic form.
RRADSTRCONST : Real root arbitrary domain structure constants.
RRADVARMATRICES : Real root arbitrary domain multiplication matrices of variables.
RRUADCOUNT : Real root univariate arbirary domain count.
RRUADOM : Real Root Univariate Arbitrary Domain Definition Module.
RRUADPOLTOVEC : Real root univariate arbitrary domain polynomial to vector.
RRUADQUADFORM : Real root univariate arbitrary domain quadratic form.
RRUADSTRCONST : Real root univariate arbitrary domain structure constants.
SetAbsFunc : Set absolute value function in domain.
SetCnstFunc : Set constant test function in domain.
SetCompFunc : Set comparison function in domain.
SetConvFunc : Set conversion function in domain.
SetDdrdFunc : Set domain descriptor read function in domain.
SetDdrdFunc : Set domain descriptor read function in domain.
SetDdwrFunc : Set domain descriptor write function in domain.
SetDdwrFunc : Set domain descriptor write function in domain.
SetDifFunc : Set difference function in domain.
SetExpFunc : Set exponential function in domain.
SetFactFunc : Set factorization function in domain.
SetFactoFunc : Set factorization with variable order optimization function in domain.
SetFIntFunc : Set from integer function in domain.
SetFIPolFunc : Set from integral polynomial function in domain.

SetGcdcFunc : Set gcd-and-cofactors function in domain.
SetGcdeFunc : Set gcd-and-lin-combination function in domain.
SetGcdFunc : Set gcd function in domain.
SetInvFunc : Set inversion function in domain.
SetInvTFunc : Set inversion test function in domain.
SetLcmFunc : Set lcm function in domain.
SetNegFunc : Set negation function in domain.
SetOneFunc : Set one test function in domain.
SetPFactFunc : Set factorization function in domain.
SetPNormFunc : Set polynomial normalform function in domain.
SetProdFunc : Set product function in domain.
SetPSpolFunc : Set polynomial S-polynomial function in domain.
SetPSqfrFunc : Set polynomial squarefree factorization function in domain.
SetPSugNormFunc : Set polynomial normal with sugar strategy normalform function in domain.
SetPSugSpolFunc : Set polynomial normal with sugar strategy S-polynomial function in domain.
SetQrFunc : Set quotient and remainder function in domain.
SetQuotFunc : Set quotient function in domain.
SetReadFunc : Set read function in domain.
SetRemFunc : Set remainder function in domain.
SetSignFunc : Set sign function in domain.
SetSumFunc : Set sum function in domain.
SetToipFunc : Set conversion-to-integer-polynomial function in domain.
SetVlddFunc : Set variable list from domain descriptor function in domain.
SetVlddFunc : Set variable list from domain descriptor function in domain.
SetWritFunc : Set write function in domain.
VdCons : Variable list and domain descriptor construct.
VdD : Variable list and domain descriptor domain descriptor part.
VdD : Variable list and domain descriptor domain descriptor part.
VdParts : Variable list and domain descriptor parts.
VdRead : Variable list and domain descriptor read.
VdV : Variable list and domain descriptor variable list part.

DOS

DOS : Call DOS program.
DOS : DOS.

double

ANPEDE : Algebraic number primitive element for a double extension.

doubly

IDEGCD : Integer doubly extended greatest common divisor algorithm.

dummy

dummycnst : Dummy constant.
dummyfactorize : Dummy factorize.

E-groebner

DIDPEGB : Distributive domain polynomial E-groebner basis.
DIIPEGB : Distributive integral polynomial E-groebner basis.

E-normal

DIDPENF : Distributive domain polynomial E-normal form.

e-normal

DIIPENF : Distributive integral polynomial e-normal form.

each

ForEachinList : For each element e in r apply function f.
ForEachinRep : For each pair (n,e) in r apply function f.

edit

EDIT : Edit file with name s.
EDIT : Edit.

element

AFDIF : Algebraic number field element difference.
AFFINT : Algebraic number field element from integer.
AFFRN : Algebraic number field element from rational number.
AFNEG : Algebraic number field element negation.
AFPAFP : Algebraic number field polynomial algebraic number field element product.
AFPAFQ : Algebraic number field polynomial algebraic number field element quotient.
AFPROD : Algebraic number field element product.
AFSUM : Algebraic number field element sum.
ANFAF : Algebraic number from algebraic number field element.
ANIIPE : Algebraic number isolating interval for a primitive element.
ANPEDE : Algebraic number primitive element for a double extension.
ANREPE : Algebraic number represent element of a primitive extension.
FFFINT : Finite field element from integer.
FFRAND : Finite field element, random.
ForEachinList : For each element e in r apply function f.
IPDECMP : integral polynomial domain element composition.
LASTEL : Last element.
LEINST : List element insertion.
LELT : List element.
LEROT : List element rotation.
LIST1 : List, 1 element.
SETADD : set add element.
SetElementP : Set element predicate.
SetElementPQ : Set element predicate equal.
SLELT : Set list element.
VDELEL : Vector delete element.
VIERED : Vector of integers, element reduction.

elementary

DIRPES : Distributive rational polynomial elementary symmetric functions.
ELEMP : Elementary Pointer.
MASELEM : MAS Elementary Functions Definition Module.

elements

LIST10 : List, 10 elements.
LIST2 : List, 2 elements.
LIST3 : List, 3 elements.
LIST4 : List, 4 elements.
LIST5 : List, 5 elements.
LIST6 : list of 6 elements.
VEL : Vector elements.

eliminate

DIDPELIMDGB : Distributive domain polynomial eliminate D-groebner base.
DIIPELIMDGB : Distributive integral polynomial eliminate D-groebner base.
FORELIMXOPS : formula eliminate extended operation symbols.
PQELIMXOPS : polynomial equation eliminate extended operation symbols.
PQELIMXOPS1 : polynomial equation eliminate extended operation symbols.

elimination

FORPREPQE : formula prepare quantifier elimination.
IMDET : Integer matrix determinant, using Gaussian elimination.
IMGE : Integer matrix Gaussian elimination.
IMGELUD : Integer matrix Gaussian elimination LU-decomposition.
MLDPREPQE : maslog demonstration prepare quantifier elimination.
RNMDET : Rational number matrix determinant, using Gaussian elimination.
RNMGE : Rational number matrix Gaussian elimination.
RNMGELUD : Rational number matrix Gaussian elimination LU-decomposition.
RQEOPTSET : real quantifier elimination options set.
RQEOPTWR : real quantifier elimination option write.
RQEPRRC : Real Quantifier Elimination with Parametric Real Root Count.
RQEQE : real quantifier elimination quantifier elimination.
RQEQE : real quantifier elimination quantifier elimination.

empty

ColEmpty : Colouring empty.
ColIsEmpty : Colouring is empty.
CondEmpty : Condition empty.
CondIsEmpty : Condition is empty.
EMPTYQUE : Empty Queue.

encode

ECENV : Encode environment.

enqueue

ENQUE : Enqueue.

enter

ENTER : Enter into symbol table.
ENTER : Enter into symbol table.
FORVTENTER : formula variable table enter.

entries

TFFTUPLE : type formula from coefficient tuple with joker entries.

environement

COPYTOENV : Copy to environement.
EXTENDENV : Extend environement.

environment

DCENV : Decode environment.
ECENV : Encode environment.
longjmp : Long jump to old environment.
setjmp : Set jump environment.

equal

ADLGeqH : Arbitrary domain polynomial list G equal H.
ADPFeqG : Arbitrary domain polynomial F equal G.
APNELD : Arbitrary precision floating point number of equal leading digits.
ASSOCQ : Associate equal.
ASSOCQ : Associate equal.
EQPLCL : Equal lists of coloured polynomials.
EQUAL : Equal.
IBeqGB : Inovlutive Base equal Groebner Base.
LSRCHQ : List search equal.
MEMQ : Membership test equal pointers.
OCCURQ : Occurs test equal pointers.
SetAddQ : Set add equal.
SetElementPQ : Set element predicate equal.
SetMinusCQ : Set minus constructive equal.
SetMinusQ : Set minus equal.
SetUnionQ : Set union equal.

equality

LEQUAL : List equality.

equation

HEQ : Homogenous Equation.
HSEQ : Homogenous System of Equation.
IEQ : Special Solution for inhomogenous commutative equation.
ISEQ : Special Solution for inhomogenous commutative system of equation.
MIPISE : Modular integral polynomial mod ideal, solution of equation.
MIUPSE : Modular integral univariate polynomial, solution of equation.
MLPQSMPL : Masload Polynomial Equation Simplify Definition Module.
NLHEQ : Non-Commutative Homogenous Equation.
NLHSEQ : Non-Commutative Homogenous System of Equation.
NLIEQ : Special Solution for inhomogenous non-commutative equation.
NLISEQ : Special Solution for inhomogenous non-commutative system of equation.
pqatom : polynomial equation atomic formula.
PQBASE : Polynomial Equation Base Definition Module.
PQBASE : symbol to mark a polynomial equation special atomic formula.
PQCnfSimplify : polynomial equation cnf based simplification.
PQDnfSimplify : polynomial equation dnf based simplification.
PQELIMXOPS : polynomial equation eliminate extended operation symbols.
PQELIMXOPS1 : polynomial equation eliminate extended operation symbols.
pqgpol : polynomial equation get polynomial.
pqgrel : polynomial equation get relation symbol.
PQIREAD : polynomial equation infix read.
pqmkaf : polynomial equation simplification make atomic formula.
PQMKCNF : polynomial equation make disjunctive normal form.
PQMKDNF : polynomial equation make disjunctive normal form.
PQMKPOS : polynomial equation make positive.
PQMKPRENEX : polynomial equation make prenex.
PQMKVD : polynomial equation make variable names disjoint.
PQOPT : polynomial equation options.
pqpaf : polynomial equation simplification parse atomic formula.
PQPPRT : polynomial equation pretty print.
PQPREAD : polynomial equation read.
PQPRING : polynomial equation polynomial ring.
PQPRINGWR : polynomial equation polynomial ring write.
pqprtaf : polynomial equation simplification print atomic formula.
pqreadaf : polynomial equation read atomic formula.
PQSCNF : polynomial equation simplification normal form.
PQSDNF : polynomial equation simplification normal form.
PQSIMPLIFY : polynomial equation simplify.
pqsimplifyaf : polynomial equation simplify atomic formula.
PQSIMPLIFYP : polynomial equation simplify.
pqsmart : polynomial equation atomic formula smart simplification.
PQSMPL : Polynomial Equation Simplification Definition Module.
PQSMPL : polynomial equation simplify.
PQTEXW : polynomial equation tex write.
pqtexwaf : polynomia equation tex write atomic formula.
SACLDIO : SAC Linear Diophantine Equation System Definition Module.
SIC : Special Solution for inhomogenous commutative system of equation.
SINL : Special Solution for inhomogenous non-commutative system of equation.
SYHC : Syzygy for homogenous commutative system of equation.
SYHNL : Syzygy for homogenous non-commutative system of equation.

error

ERROR : Error.
ErrorHandler : Error handler.
EStreamKind : Error stream kind.
MASERR : error indicators.
MASERR : MAS Error Definition Module.

ev

MAKERN : UGB rational exponent vector list from integer ev list.
SKPRO2 : UGB rational exponent vector scalar product with integer ev.

evaluation

AFPEMV : Algebraic number field polynomial evaluation of main variable.
AFPEV : Algebraic number field polynomial evaluation.
AFPME : Algebraic number field, polynomial multiple evaluation.
DIIPEM : Distributive integral polynomial evaluation of main variable.
DIIPEV : Distributive integral polynomial evaluation of the i-th variable.
DIRPEM : Distributive rational polynomial evaluation of main variable.
DIRPEV : Distributive rational polynomial evaluation of the i-th variable.
IPAFME : Integral polynomial, algebraic number field multiple evaluation.
IPCEVP : Integral polynomial, choice of evaluation points.
IPEMV : Integral polynomial evaluation of main variable.
IPEVAL : Integral polynomial evaluation.
IUPBEI : Integral univariate polynomial binary rational evaluation, integer output.
IUPBES : Integral univariate polynomial binary rational evaluation of sign.
IUPBRE : Integral univariate polynomial binary rational evaluation.
MMPEV : Matrix of modular polynomials evaluation.
MPEMV : Modular polynomial evaluation of main variable.
MPEVAL : Modular polynomial evaluation.
RPAFME : Rational polynomial, algebraic number field multiple evaluation.
RPEMV : Rational polynomial evaluation, main variable.

evaluator

EVALUATE : Lisp evaluator.

even

IEVEN : Integer even.
MASEVEN : Even.

evord

EvordPop : evord pop.
EvordPush : evord push.
EvordReset : Reset evord.

EVORD

EvordSet : EVORD set.

evord

EvordWrite : Evord Write.

exact

MAIPDE : Matrix of integral polynomials determinant, exact division algorithm.

exchange

DIPEXC : Distributive polynomial exchange variables.
EVEXC : Exponent vector exchange.

exclude

EX0PL : Exclude 0 from PL.
EXPPL : Exclude P from GB.

exec

DIMEXE : Parametric dimension exec.
EXECRD : Exec read.
NFEXEC : Parametric ideal membership exec.

execute

CGBOUT : Comprehensive-Groebner-Basis execute and output.
EXEUGB : UGB execute.
UGBBIN : UGB input, execute and output.

execution

EXECRD : UGB execution options read.

existence

ADINVT : Arbitrary domain inverse existence test.

expansion

IMDETL : Integer matrix determinant, using Laplace expansion.
RNMDETL : Rational number matrix determinant, using Laplace expansion.

explode

EXPLOD : Explode symbol.
EXPLOD : Explode symbol.

exponenetvector-Check

NLSPCEGB : Non-Commutative S-Polynomials with Coefficients and Exponenetvector-Check
SPCEGB : S-Polynomials with Coefficients and Exponenetvector-Check for Groebner Base.

exponent

APEXPT : Arbitrary precision floating point exponent.
CP2 : UGB linear form product with rational exponent vector list 2.
CQ2 : UGB linear form product with rational exponent vector list.
DEGRE : UGB total degree of a list of rational exponent vectors.
DIFF : UGB difference set for rational exponent vector list.
DIFF1 : UGB difference set for two rational exponent vector list.
DIIRAS : Distributive integral polynomial random sparse exponent vector.
DILFEL : Distributive polynomial list from exponent vector list.
DIPEVL : Distributive polynomial exponent vector leading monomial.
DIPEVP : Distributive polynomial exponent vector product.
DIRRAS : Distributive rational polynomial, random sparse exponent vector.
DMEVAD : Degree matrix exponent vector add.
DO1 : UGB add last component to exponent vector.
EDIPEVL : Extended distributive polynomial exponent vector of the leading monomial.
EPREAD : Exponent read.
EVASC : Exponent vector ascending.
EVCADD : Exponent vector component add.
EVCNSTR : exponent vector constant relatively.
EVCOMP : Exponent vector compare.
EVCOMP : UGB exponent vector compare.
EVCSUB : Exponent vector component subtract.
EVDEL : Exponent vector delete.
EVDER : Exponent vector derivation.
EVDFSI : Exponent vector difference and sign.
EVDIF : Exponent vector difference.
EVDIF2 : Exponent vector difference.
EVDOV : Exponent vector dependency on variables.
EVEXC : Exponent vector exchange.
EVEXT : exponent vector extension.
EVF : Exponent Vector First.
EVGBIT : Exponent vector groebner base intersection test.
EVGCD : Exponent vector greatest common divisor.
EVIGLC : Exponent vector inverse graded lexicographical compare.
EVILCI : Exponent vector inverse lexicographical compare inverse exponent vector.
EVILCI : Exponent vector inverse lexicographical compare inverse exponent vector.
EVILCP : Exponent vector inverse lexicographical compare.
EVINV : Exponent vector introduction of new variables.
EVITDC : Exponent vector inverse total degree compare.
EVL : Exponent Vector Length.
EVLCM : Exponent vector least common multiple.
EVLFCP : Exponent vector linear form compare.
EVLFCP : UGB exponent vector linear form compare.
EVLGIL : Exponent vector list generate for inverse lexicographical sequence.
EVLGTD : Exponent vector list generate for total degree.
EVLINV : Exponent vector list introduction of new variables.
EVLRNBSO : Rational exponent vector list bubble sort.
EVMT : Exponent vector multiple test.
EVMTJ : Exponent vector multiple test in the sense of Janet.
EVNNZE : Exponent vector number of non zero exponents.

EVOWRITE : Exponent vector order write.
EVPLM : Exponent vector pair-list merge.
EVPLSO : Exponent vector pair-list sort.
EVR : Exponent Vector Reduction.
EVRAND : Exponent vector random.
EVRASP : Exponent vector random.
EVRNC : Rational exponent vector compare.
EVRNGL : Rational exponent vector inverse graded lexicographical compare.
EVRWTDEG : Exponent vector rational-weighted total degree.
EVSIGN : Exponent vector signum.
EVSSPROD : Exponent vektor set sorted product.
EVSU : Exponent vector substitution.
EVSUM : Exponent vector sum.
EVT : Exponent Vector Test.
EVTDEG : Exponent vector total degree.
EVTSZ : Exponent vector test if starting with i zero exponents.
EVUMSORT : Exponent vector unique merge sort.
EVZERO : Exponent vector zero.
EXPTU : UGB extract exponent vector list from polynomial list.
GBE : Groebner Base with Exponent Vector Check.
GBEF : Groebner Base with Exponent Vector Check and Factors.
MAKERN : UGB rational exponent vector list from integer ev list.
MKSET : UGB rational exponent vector list difference list.
NEWDIF : UGB exponent vector list difference from polynomials.
NLGBE : Non-Commutative Groebner Base with Exponent Vector Check.
NLGBEF : Non-Commutative Groebner Base with Exponent Vector Check and Factors.
PDIF : UGB rational exponent vector list difference list, incremental.
RNVDIF : UGB rational exponent vector difference.
SCMULT : UGB rational exponent vector rational number product.
SCPROD : UGB rational exponent vector scalar product.
SKPRO2 : UGB rational exponent vector scalar product with integer ev.
SYGBE : Syzygy for Groebner Base with Exponent Vector.

exponential

EXPF : Exponential.
MASEXP : Exponential function.
SetExpFunc : Set exponential function in domain.

exponentiation

ADEXP : Arbitrary domain exponentiation.
APEXP : Arbitrary precision floating point exponentiation.
CEXP : Complex number exponentiation.
DIIPEX : Distributive integral polynomial exponentiation.
DINPEX : Distributive non-commutative polynomial exponentiation.
DIPEXP : Distributive polynomial exponentiation.
DIRPEX : Distributive rational polynomial exponentiation.
FEXP : Floating point exponentiation.
FFEXP : Finite field exponentiation.
IEXP : Integer exponentiation.
IPEXP : Integral polynomial exponentiation.
MDEXP : Modular digit exponentiation.

MIEXP : Modular integer exponentiation.
MPEXP : Modular polynomial exponentiation.
OEXP : Octonion number exponentiation.
QEXP : Quaternion number exponentiation.
RFEXP : Rational function exponentiation.
RNEXP : Rational number exponentiation.
RPEXP : Rational polynomial exponentiation.

exponents

EVNNZE : Exponent vector number of non zero exponents.
EVTSZ : Exponent vector test if starting with i zero exponents.

expr

DEFE : Define expr function.

expression

GENTE : Generate typed expression.
LAMBDAP : Test if expression S is a lambda form.
SPECIALFORM : Test if expression S is a special form.

extend

CPEXTEND : Critical pair extend.
DIDPLEXTAL : Distributive domain polynomial list extend array list.
DIIPLEXTAL : Distributive integral polynomial list extend array list.
EXTENDENV : Extend environement.
LDIPEXTEND : List of distributive polynomials extend.

extende

ADEPtriple : Arbitrary domain extende polynomial triple.

extended

ADDNFEDIP : Arbitrary domain domain number from extended distributive polynomial.
ADEPCompose : Arbitrary domain extended polynomial compose.
ADEPCrumble : Arbitrary domain extended polynomial crumble.
ADEPdegree : Arbitrary domain extended polynomial degree.
ADEPheadterm : Arbitrary domain extended plynomial head-term.
ADEPmark : Arbitrary domain extended polynomial mark.
ADEPNFJ : Arbitrary domain extended polynomial normalform in the sense of Janet.
ADEPpolynomial : Arbitrary domain extended polynomial polynomial.
ADEPuntriple : Arbitrary domain extended polynomial untriple.
DEGCD : Digit extended greatest common divisor.
DILEBBS : Distributive List Extended Bubble Sort.
DILEP2P : Distributive polynom list extended polynom to polynom.
DIPEINFJ : Integral distributive extended polynomial normal form in the sense of Janet.
DIPENFJ : Distributive extended polynomial normal form in the sense of Janet.
DIREGB : Distributive rational polynomials extended groebner basis.

ECPINSERT : Extended critical pair insertion.
ECPLCMHT : Extended critical pair select the least common multiple of head terms.
ECPPOLY1 : Extended critical pair select the first extended distributive polynomial.
ECPPOLY1 : Extended critical pair select the first extended distributive polynomial.
ECPPOLY2 : Extended critical pair select the second extended distributive
ECPPOLY2 : Extended critical pair select the second extended distributive
ECPSELECT : Select an extended critical pair from the extended critical pair list.
ECPSELECT : Select an extended critical pair from the extended critical pair list.
ECPSUGAR : Extended critical pair select sugar.
ECPUNEXTEND : Extended critical pair un-extend.
ECPWRITE : Extended critical pair write.
EDIIFSUGNF : Extended distributive integral function polynomial normal with sugar
EDIIFSUGSP : Extended distributive integral function polynomial normal with sugar
EDIPEVL : Extended distributive polynomial exponent vector of the leading monomial.
EDIPNOR : Extended distributive polynomial normal form.
EDIPSP : Extended distributive polynomial S-polynomial.
EDIPSUGAR : Extended distributive polynomial sugar.
EDIPSUGCON : Extended distributive polynomial normal with sugar strategy constructor.
EDIPSUGNOR : Extended distributive polynomial normal with sugar strategy normal form.
EDIPSUGSP : Extended distributive polynomial normal with sugar strategy S-polynomial.
EDIPUNEXTEND : Extended distributive polynomial un-extend.
EDIPWRITE : Extended distributive polynomial write.
FORELIMXOPS : formula eliminate extended operation symbols.
IDEGCD : Integer doubly extended greatest common divisor algorithm.
IEGCD : Integer extended greatest common divisor algorithm.
LECPUNEXTEND : List of extended critical pairs un-extend.
LECPWRITE : List of extended critical pairs write.
LEDIPUNEXTEND : List of extended distributive polynomials un-extend.
LEDIPWRITE : List of extended distributive polynomials write.
MUPEGC : Modular univariate polynomial extended greatest common divisor.
PQELIMXOPS : polynomial equation eliminate extended operation symbols.
PQELIMXOPS1 : polynomial equation eliminate extended operation symbols.
RUPEGC : Rational univariate polynomial extended greatest common divisor.
SetDIPAGBStrategy : Set the DIPAGB strategy option for the extended critical pair selection.
SetECPInsert : Set the extended critical pair insertion function.
SetECPSelect : Set the extended critical pair selection procedure.
SetECPWrite : Set the extended critical pair write procedure.
SetEDIPNormalform : Set the extended distributive polynomial normalform function.
SetEDIPSPolynomial : Set the extended distributive S-polynomial function.
SetEDIPUnExtend : Set the extended distributive polynomial un-extension function.
SetEDIPWrite : Set the extended distributive polynomial write procedure.
TIPRNGB : DIP Rational Extended Groebner Bases Definition Module.
UPDATE : Update of extended ideal basis and extended critical pair list as required
UPDATE : Update of extended ideal basis and extended critical pair list as required
WriteDIPAGBStrategy : Write the DIPAGB strategy option for the extended critical pair selection

extension

ANPEDE	: Algebraic number primitive element for a double extension.
ANREPE	: Algebraic number represent element of a primitive extension.
DIGFET	: DIP G base successful extension test.
DIGISM	: DIP G base index search for extension multiple univariats.
DIGISR	: DIP G base index search for extension reductas.
DINTFE	: DIP normalized tupel field extension.
DIPADGBext	: distributive polynomial arbitrary domain groebner basis extension.
DIPEXTEND	: Distributive polynomial extension.
EVEXT	: exponent vector extension.
RRVTEXT	: Real root vector of tupels extension.
SetCPExtend	: Set the critical pair extension function.
SetDIPExtend	: Set the distributive polynomial extension function.

extensions

SACEXT1	: SAC Extensions 1 Definition Module.
SACEXT2	: SAC Extensions 2 Definition Module.
SACEXT3	: SAC Extensions 3 Definition Module.
SACEXT4	: SAC Extensions 4 Definition Module.
SACEXT5	: SAC Extensions 5 Definition Module.
SACEXT6	: SAC Extensions 6 Definition Module.
SACEXT7	: SAC Extensions 7 Definition Module.
SACEXT8	: SAC Extensions 8 Definition Module.
ZULFO	: UGB find admissible extensions of linear forms.

extent

EXTENT	: Extent.

exterior

DIPE	: DIP Exterior Algebra Definition Module.
EIMWRT	: Exterior integral matrix write.
EIVABS	: Exterior integral vector absolute value.
EIVAPP	: Exterior integral vector absolute primitive part.
EIVCPP	: Exterior integral vector content and primitive part.
EIVEPR	: Exterior integral vector exterior product.
EIVEPR	: Exterior integral vector exterior product.
EIVFUP	: Exterior integral vector from univariate integral polynomial
EIVILP	: Exterior integral vector inner left product.
EIVIP	: Exterior integral vector integer product.
EIVIQ	: Exterior integral vector integer quotient.
EIVIRP	: Exterior integral vector inner right product.
EIVNEG	: Exterior integral vector negative.
EIVPP	: Exterior integral vector primitive part.
EIVSIG	: Exterior integral vector sign.
EIVSUM	: Exterior integral vector sum.
EIVWRT	: Exterior integral vector write.
EXIDET	: Exterior integral matrix determinant.
EXIDT2	: Exterior integral matrix determinant 2.
EXMHOM	: Exterior matrix homomorphism.
EXVHOM	: Exterior vector homomorphism.
ILEXPR	: Index list exterior product.

POWSEV : Power of variable symmetric product with exterior vector.
UIPSIL : Univariate integral polynomial symmetric product with exterior index list.
UIPSIV : Univariate integral polynomial symmetric product with exterior integral vector.

external

InitExternals : Initialize external compiled procedures.
InitExternalsA : Initialize external compiled arithmetic procedures.
InitExternalsB : Initialize external compiled polynomial procedures.
InitExternalsC : Initialize external compiled non-commutative polynomial procedures.
InitExternalsD : Initialize external compiled ideal decomposition and root procedures.
InitExternalsE : Initialize external compiled arbitrary domain procedures.
InitExternalsG : Tell Modula and LISP about external compiled procedures.
InitExternalsI : Initialize external compiled interface procedures.
InitExternalsJ : Initialize external compiled arithmetic procedures.
InitExternalsL : Initialize external compiled linear algebra procedures.
InitExternalsM : Initialize external compiled real root procedures.
InitExternalsML : Initialze external compiled logic procedures.
InitExternalsPQSMPL : initialize external compiled PQS-procedures.
InitExternalsQ : Initialize external compiled arithmetic Q procedures.
InitExternalsS : Tell Modula and LISP about external compiled procedures.
InitExternalsU : Initialize external compiled utility procedures.

externals

InitExternalsMLDEMO : Initialize externals maslog demonstration procedures.

extra

ADEXTRA : Arbitrary domain extra definition module.

extract

DIPXCM : distributive polynomial extract constant monomials.
EXPTU : UGB extract exponent vector list from polynomial list.

extraneous

GLEXTP : Global extraneous polynomials remove.
REXTP : Remove extraneous polynomials.

f0

Compiledf0 : Compiled function declaration f0.

f1

Compiledf1 : Compiled function declaration f1.

f2

Compiledf2 : Compiled function declaration f2.
FORAPPLYATF2 : formula apply to atomic formula f2.

f3

Compiledf3 : Compiled function declaration f3.

f4

Compiledf4 : Compiled function declaration f4.

f5

Compiledf5 : Compiled function declaration f5.

factor

BGFUP : Base Generators Factor Update.
DIPRLF : distributive polynomials reduce list of polynomials with factor.
IPFCB : Integral polynomial factor coefficient bound.
IPFLC : Integral polynomial factor list combine.
IPGFCB : Integral polynomial Gelfond factor coefficient bound.
IUPFDS : Integral univariate polynomial factor degree set.
NLBGFUP : Non-Commutative Base Generators Factor Update.
NOEPIP : Noether Polynomial Factor Multiplication.

factor-Sugar

SetFacSugar : Set Factor-Sugar for procedure GroebnerBases1:

factorial

IFACTL : Integer factorial.

factorisation

DIRFAC : Distributive rational polynomial factorisation.

factorization

ADFACT : Arbitrary domain factorization.
ADFACTO : Arbitrary domain factorization with variable order optimization.
ADPFACT : Arbitrary domain polynomial factorization.
ADPSFF : Arbitrary domain polynomial squarefree factorization.
AFUPSF : Algebraic number field univariate polynomial squarefree factorization.
DIPFAC : distributive polynomial factorization.
DIPSFF : distributive polynomial squarefree factorization.
GBSYSF : Groebner system with factorization.
GSYSF : Groebner system with factorization.
IFACT : Integer factorization.
IPFAC : Integral polynomial factorization.
IPFSD : Integral polynomial factorization, second derivative.
IPSF : Integral polynomial squarefree factorization.
IPSFF : integral polynomial squarefree factorization
IPSFSD : Integral squarefree factorization, second derivative.
ISFPF : Integral squarefree polynomial factorization.
IUPFAC : Integral univariate polynomial factorization.
IUSFPF : Integral univariate squarefree polynomial factorization.

MUPDDF : Modular univariate polynomial distinct degree factorization.
MUPFS : Modular univariate polynomial factorization, special.
MUPSFF : Modular univariate polynomial squarefree factorization.
SACMUFAC : SAC Modular Univariate Polynomial Factorization Definition Module.
SACPFAC : SAC Polynomial Factorization Definition Module.
SACPRIM : SAC Factorization and Prime Number Definition Module.
SACUPFAC : SAC Univariate Polynomial Factorization Definition Module.
SetFactFunc : Set factorization function in domain.
SetFactoFunc : Set factorization with variable order optimization function in domain.
SetPFactFunc : Set factorization function in domain.
SetPSqfrFunc : Set polynomial squarefree factorization function in domain.

factorization-Berlekamp

MUPFBL : Modular univariate polynomial factorization-Berlekamp algorithm.

factorize

dummyfactorize : Dummy factorize.

factorlist

IPFLMER : integral polynomial factorlist merge.

factors

FINDCP : Find white factors.
GBEF : Groebner Base with Exponent Vector Check and Factors.
GBF : Groebner Base with Factors.
NLGBEF : Non-Commutative Groebner Base with Exponent Vector Check and Factors.
NLGBF : Non-Commutative Groebner Base with Factors.
NORMF : Normative Factors.

family

WRUGF : Write universal Groebner family.

fermat

FRESL : Fermat residue list.
FRLSM : Fermat residue list, single modulus.

fexpr

DEFF : Define fexpr function.

field

AFCOMP : Algebraic number field comparison.
AFDIF : Algebraic number field element difference.
AFFINT : Algebraic number field element from integer.
AFFRN : Algebraic number field element from rational number.
AFINV : Algebraic number field inverse.
AFNEG : Algebraic number field element negation.
AFPAFP : Algebraic number field polynomial algebraic number field element product.
AFPAFP : Algebraic number field polynomial algebraic number field element product.
AFPAFQ : Algebraic number field polynomial algebraic number field element quotient.
AFPAFQ : Algebraic number field polynomial algebraic number field element quotient.
AFPBRI : Algebraic number field polynomial basis real root isolation.
AFPCLL : Algebraic number field polynomial real root isolation, Collins-Loos
AFPDIF : Algebraic number field polynomial difference.
AFPDMV : Algebraic number field polynomial derivative, main variable.
AFPEMV : Algebraic number field polynomial evaluation of main variable.
AFPEV : Algebraic number field polynomial evaluation.
AFPFIP : Algebraic number field polynomial from integral polynomial.
AFPFRP : Algebraic number field polynomial from rational polynomial.
AFPINT : Algebraic number field polynomial integration.
AFPME : Algebraic number field, polynomial multiple evaluation.
AFPMON : Algebraic number field polynomial monic.
AFPMPR : Algebraic number field polynomial minimal polynomial of a real root.
AFPNEG : Algebraic number field polynomial negative.
AFPNIP : Algebraic number field polynomial normalize to integral polynomial.
AFPPR : Algebraic number field polynomial product.
AFPQR : Algebraic number field polynomial quotient and remainder.
AFPRCL : Algebraic number field polynomial real root isolation, Collins-Loos algorithm.
AFPRII : Algebraic number field polynomial real root isolation induction.
AFPRLS : Algebraic number field polynomial real root list separation.
AFPROD : Algebraic number field element product.
AFPRRI : Algebraic number field polynomial relative real root isolation.
AFPRRS : Algebraic number field polynomial real root separation.
AFPSUM : Algebraic number field polynomial sum.
AFQ : Algebraic number field quotient.
AFSIGN : Algebraic number field sign.
AFSUM : Algebraic number field element sum.
AFSUPB : Algebraic number field squarefree univariate polynomial squarefree
AFUPBA : Algebraic number field univariate polynomial squarefree basis
AFUPCB : Algebraic number field univariate polynomial coarsest squarefree basis.
AFUPGC : Algebraic number field univariate polynomial greatest common divisor
AFUPGS : Algebraic number field polynomial greatest squarefree divisor.
AFUPRB : Algebraic number field univariate polynomial root bound.
AFUPRL : Algebraic number field polynomial, root of a linear polynomial.
AFUPSF : Algebraic number field univariate polynomial squarefree factorization.
AFUPSR : Algebraic number field univariate polynomial, sign at a rational
ANFAF : Algebraic number from algebraic number field element.

finite

DOMFF : MAS Domain Finite Field Definition Module.
DomLoadFF : Domain load finite field.
FFCOMP : Finite field comparison.
FFDIF : Finite field difference.
FFEXP : Finite field exponentiation.
FFFINT : Finite field element from integer.
FFHOM : Finite field homomorpism.
FFINV : Finite field inverse.
FFNEG : Finite field negation.
FFONE : Finite field one.
FFPROD : Finite field product.
FFQ : Finite field quotient.
FFRAND : Finite field element, random.
FFREAD : Finite field read.
FFSUM : Finite field sum.
FFWRITE : Finite field write.
MASFF : MAS Finite Field Definition Module.

first

DIPFIRST : Distributive polynomial first polynomial,
ECPPOLY1 : Extended critical pair select the first extended distributive polynomial.
EVF : Exponent Vector First.
FIRST : First.
FIRST2 : First 2.
FIRST3 : First 3.
FIRST4 : First 4.
INITUPDATE : The initialization function as a first call of UPDATE.
SFIRST : Set first.

flag

SetDIPAGBOptions : Set the trace flag, the strategy and the variable weight list of the
SetDIPAGBTraceFlag : Set the DIPAGB trace flag.
WriteDIPAGBOptions : Write the current trace flag, strategy and variable weight list of the
WriteDIPAGBTraceFlag: Write the DIPAGB trace flag in the output stream.

floating

APABS : Arbitrary precision floating point absolute value.
APCMPR : Arbitrary precision floating point compare.
APCOMP : Arbitrary precision floating point composition.
APDIFF : Arbitrary precision floating point difference.
APEXP : Arbitrary precision floating point exponentiation.
APEXPT : Arbitrary precision floating point exponent.
APFINT : Arbitrary precision floating point from integer.
APFRN : Arbitrary precision floating point from rational number.
APLG10 : Arbitrary precision floating point logarithm base 10.
APMANT : Arbitrary precision floating point mantissa.
APNEG : Arbitrary precision floating point negative.
APNELD : Arbitrary precision floating point number of equal leading digits.
APPI : Arbitrary precision floating point pi.

APPROD : Arbitrary precision floating point product.
APQ : Arbitrary precision floating point quotient.
APROOT : Arbitrary precision floating point n-th root.
APSHFT : Arbitrary precision floating point shift.
APSIGN : Arbitrary precision floating point sign.
APSPRE : Arbitrary precision floating point set precision.
APSUM : Arbitrary precision floating point sum.
APWRIT : Arbitrary precision floating point write.
DOMAPF : MAS Domain Arbitrary Precision Floating Point Definition Module.
DomLoadAPF : Domain load arbitrary precision floating point.
FEXP : Floating point exponentiation.
FFGI : Floating point from gamma integer.
FFINT : Floating point from integer.
FFRN : Floating point from rational number.
FLOG10 : Floating point logarithm base 10.
IFF : Integer from floating point.
MASAPF : MAS Arbitrary Precision Floating Point Definition Module.
MASF : MAS Floating Point Definition Module.
RNFAP : Rational number from arbitrary precision floating point.
RNFF : Rational number from floating point.

floor

IFCL2 : Integer, floor and ceiling, logarithm, base 2.
RNFCL2 : Rational number floor and ceiling of logarithm, base 2.
RNFLOR : Rational number, floor of.

for

ForEachinList : For each element e in r apply function f.
ForEachinRep : For each pair (n,e) in r apply function f.

foreign

clock : clock Foreign Module.
maskpathsea : kpathsearch Foreign Module.
masreadline : readline Foreign Module.
massig : MAS Signal Handling Foreign Module.
setjmp : Setjmp Foreign Module.

form

ALFA : Automatic Linear Form Adaption.
ALFRA : Automatic Linear Form Readaption.
CLF2 : UGB compute linear form from difference set 2.
CLF3 : UGB compute linear form from difference set 3.
COMPLF : UGB compute linear form from difference set.
CP2 : UGB linear form product with rational exponent vector list 2.
CQ2 : UGB linear form product with rational exponent vector list.
CSPUR : UGB trace for linear form 2.
DIDPDNF : Distributive domain polynomial D-normal form.
DIDPENF : Distributive domain polynomial E-normal form.
DIIFNF : Distributive integral function polynomial normal form.
DIIPDNF : Distributive integral polynomial normal form.

DIIPENF : Distributive integral polynomial e-normal form.
DIIPNF : Distributive integral polynomial normal form.
DIIPNFJ : Distributive integral polynomial normal form in the sense of Janet.
DILADNF : distributive polynomial list arbitrary domain normal form.
DINLNF : Distributive non-commutative polynomial left normal form.
DIPADNF : distributive polynomial arbitrary domain normal form.
DIPEINFJ : Integral distributive extended polynomial normal form in the sense of Janet.
DIPENFJ : Distributive extended polynomial normal form in the sense of Janet.
DIPINFJ : Integral Distributive polynomial normal form in the sense of Janet.
DIPINFJS : Integral Distributive polynomial normal form in the sense of Janet modulo a
DIPNFJ : Distributive polynomial normal form in the sense of Janet.
DIPNFJS : Distributive polynomial normal form in the sense of Janet modulo a set of
DIPNOR : Distributive polynomial normal form.
DIRPNF : Distributive rational polynomial normal form.
DIRPNFJ : Distributive rational polynomial normal form in the sense of Janet.
EDIPNOR : Extended distributive polynomial normal form.
EDIPSUGNOR : Extended distributive polynomial normal with sugar strategy normal form.
EVLFCP : Exponent vector linear form compare.
EVLFCP : UGB exponent vector linear form compare.
ISNEUL : UGB new linear form test.
LAMBDAP : Test if expression S is a lambda form.
LF : UGB linear form.
LFCHECK : Linear form check.
LFGET : UGB get linear form from list of linear forms.
MLDMKCNF : maslog demonstration make disjunctive normal form.
MLDMKDNF : maslog demonstration make disjunctive normal form.
PFIDNOR : Integral Polynomial D Normal Form.
PFILDNOR : Integral Polynomial List D-Normal Form.
PFILNOR : Integral Polynomial List Normal Form.
PFINOR : Integral Polynomial Normal Form.
PKEGEL : UGB trace for linear form.
PLF : UGB linear form with precomputed linear forms.
PQMKCNF : polynomial equation make disjunctive normal form.
PQMKDNF : polynomial equation make disjunctive normal form.
PQSCNF : polynomial equation simplification normal form.
PQSDNF : polynomial equation simplification normal form.
RRADNFORM : Real root arbitrary domain normal form.
RRADQUADFORM : Real root arbitrary domain quadratic form.
RRINFORM : Real root integral normal form.
RRIQUADFORM : Real root integral quadratic form.
RRUADQUADFORM : Real root univariate arbitrary domain quadratic form.
RRUIQUADFORM : Real root univariate integral quadratic form.
SimplifyNf : simplify normal form.
SPECIALFORM : Test if expression S is a special form.

formatted

FILWRITE : Formatted indented list write.
FLWRITE : Formatted list write.

forms

ALLELN : UGB all linear forms from stack of projections.
ALLLF : UGB all linear forms from stack of projections and print.
LFALL : UGB all linear forms from stack of projections 1.
LFGET : UGB get linear form from list of linear forms.
MKLF1 : UGB make new linear forms 1.
MKLF2 : UGB make new linear forms 2.
MKLF3 : UGB make new linear forms 3.
NEULF : UGB compute new linear forms from new terms.
NEWL : UGB update linear forms from new terms.
NONEWL : UGB update linear forms without new terms.
PLF : UGB linear form with precomputed linear forms.
PUGB : Universal Groebner base with precomputed linear forms.
ZULFO : UGB find admissible extensions of linear forms.

formula

CGBQFF : Comprehensive Groebner basis quantifier free formula.
ComputeTypeFormula : compute type formula.
CPART : Comprehensive-Groebner-Basis quantifier free formula.
FdCons : Formula and dimension construct.
FdD : Formula and dimension formula part.
FdD : Formula and dimension formula part.
FdF : Formula and dimension formula part.
FdF : Formula and dimension formula part.
FdParts : Formula and dimension parts.
FdV : Formula and dimension variable list part.
FdWrite : Formula and dimension write.
FORAPPLYAT : formula apply to atomic formular.
FORAPPLYATF2 : formula apply to atomic formula f2.
FORAPPLYATF2 : formula apply to atomic formula f2.
FORCONTBDVAR : formula contain bound variable.
FORCONTVAR : formula contain variable.
FORCOUNTAF : formula count atomic formulas.
FORELIMXOPS : formula eliminate extended operation symbols.
FORGARGS : formula get arguments.
FORGLVAR : formula get list of variables.
FORGOP : formula get operation.
FORIMQB : formula innermost quantifier block.
FORIREAD : formula infix read.
FORISATOM : formula is atomic formula.
FORISATOM : formula is atomic formula.
FORISBBFOR : formula is black-box formula.
FORISBBFOR : formula is black-box formula.
FORISBOOLVAR : formula is boolean variable.
FORISLVAR : formula is variable list.
FORISVAR : formula is variable.
FormFCond : Formula from Condition.
FORMKBINOP : formula make binary operation.
FORMKCNF : formula make cnf.
FORMKCNST : formula make constant, i.
FORMKDNF : formula make dnf.
FORMKFOR : formula make formula.

FORMKFOR : formula make formula.
FORMKLVAR : formula make list of variables.
FORMKPOS : formula make positive.
FORMKPRENEX : formula make prenex.
FORMKPRENEX1 : formula make prenex 1.
FORMKPRENEXI : formula make prenex.
FORMKQUANT : formula make quantifier.
FORMKUNOP : formula make unary operation.
FORMKVAR : formula make variable.
FORMKVD : formula make variable names disjoint.
FORPARGS : formula parse arguments.
FORPBINOP : formula parse binary operation.
FORPBINOPA : formula parse binary operation argument.
FORPFOR : formula parse formula.
FORPFOR : formula parse formula.
FORPLVAR : formula parse list of variables.
FORPPLVAR : formula print lvar.
FORPPRT : formula pretty print.
FORPPVAR : formula pretty print variable.
FORPQUANT : formula parse quantifier.
FORPQUANTA : formula parse quantifier arguments.
FORPREAD : formula prefix read.
FORPREPQE : formula prepare quantifier elimination.
FORPUNOP : formula parse unary operation.
FORPUNOPA : formula parse unary operation argument.
FORPVAR : formula parse variable.
FORPVARA : formula parse variable arguments.
FORRDLVAR : formula read list of variables.
FORRDVAR : formula read variable.
FORREPAFS : formula replace atomic formulas.
FORSIMPLIFY : formula simplify.
FORSIMPLIFYP : formula simplify prune.
FORSMPL : formula simplify.
FORSUBSTVAR : formula substitute variable.
FORTEXW : formula tex write.
FORTEXWVAR : formula tex write variable.
FORTST : formula test.
FORVTENTER : formula variable table enter.
FORVTGET : formula variable table get.
FORVTRESTORE : formula variable table restore.
FORVTSTORE : formula variable table store.
InitBbfParser : Initialize black-box formula parser.
MLDMKATOM : maslog demonstration make atomic formula.
PdF : Parametric dimension formula and dimension list part.
pqatom : polynomial equation atomic formula.
PQBASE : symbol to mark a polynomial equation special atomic formula.
pqmkaf : polynomial equation simplification make atomic formula.
pqnegaf : negate atomic formula.
pqpaf : polynomial equation simplification parse atomic formula.
pqprtaf : polynomial equation simplification print atomic formula.
pqreadaf : polynomial equation read atomic formula.
pqsimplifyaf : polynomial equation simplify atomic formula.
pqsmart : polynomial equation atomic formula smart simplification.

formular

formulas

fourth

free

from

ADDDFDIL : arbitrary domain domain descriptor from distributive polynomial list.
ADDDFDILD : arbitrary domain domain descriptor from distributive polynomial list
ADDDFDIP : arbitrary domain domain descriptor from distributive polynomial.
ADDDFDIPD : arbitrary domain domain descriptor from distributive polynomial or default.
ADDDFSTR : arbitrary domain domain descriptor from string.
ADDNFDIL : arbitrary domain domain number from distributive polynomial list.
ADDNFDILD : arbitrary domain domain number from distributive polynomial list
ADDNFDIP : arbitrary domain domain number from distributive polynomial.
ADDNFDIPD : arbitrary domain domain number from distributive polynomial or default.
ADDNFEDIP : Arbitrary domain domain number from extended distributive polynomial.
ADFI : Arbitrary domain from integer.
ADFIP : Arbitrary domain from integral polynomial.
ADPFDIP : arbitrary domain polynomial from distributive polynomial.
ADRFFADIP : arbitrary domain rational function from arbitrary domain intgeral
ADVLDD : variable list from domain descriptor.
AFFINT : Algebraic number field element from integer.
AFFRN : Algebraic number field element from rational number.
AFPFIP : Algebraic number field polynomial from integral polynomial.
AFPFRP : Algebraic number field polynomial from rational polynomial.
ALLELN : UGB all linear forms from stack of projections.
ALLLF : UGB all linear forms from stack of projections and print.
ANFAF : Algebraic number from algebraic number field element.
APFINT : Arbitrary precision floating point from integer.
APFRN : Arbitrary precision floating point from rational number.
CGBFGSYS : Comprehensive Groebner basis from Groebner system.
CGBFRM : Comprehensive-Groebner-Basis from coloured basis.
CINT : Complex number from integer.
CLF2 : UGB compute linear form from difference set 2.
CLF3 : UGB compute linear form from difference set 3.
CLISTFA : character list from atom.
ColConsCond : Colouring construct from condition.
ColpConsCond : Coloured polynomial construct from condition.
COMPLF : UGB compute linear form from difference set.
CRN : Complex number from rational number.
CRNP : Complex number from pair of rational numbers.
CSFPAR : Characteristic set from partition.
DIFIP : Distributive polynomial from distributive integral polynomial.
DIFPF : Distributive polynomial with arbitrary domain coefficients from
DIIFRP : Distributive integral polynomial from rational polynomial.
DIILFR : Distributive integral polynomial list from rational polynomial list.
DIILFRCD : DIP integral list from DIP rational list using common denominator.
DIITNT : Distributive polynomial system interval tupel from norm tupel.
DILFDILP : distributive polynomial list from distributive polynomial list over
DILFEL : Distributive polynomial list from exponent vector list.
DILFPFL : Distributive polynomial list with arbitrary domain coefficients from
DILFPL : Distributive polynomial list from polynom list.
DILPFDIL : distributive polynomials over polynomial ring list from distributive
DINTZS : DIP nomalized tupels from system zero.
DIPFADIP : distributive polynomial from arbitrary domain integral polynomial.

DIPFDIPP : distributive polynomial from distributive polynomial over polynomial ring.
DIPFIP : distributive polynomial from integral polynomial.
DIPFMO : Distributive polynomial from monomial.
DIPFP : Distributive polynomial from polynomial.
DIPLFPFL : Distributive polynomial list from recursive polynomial
DIPPFDIP : distributive polynomial over polynomial ring from distributive polynomial.
DIRFIP : Distributive rational polynomial from integral polynomial.
DIRPFT : Distributive rational polynomial from term.
DITFZS : DIP tupel from zero set.
DPFP : Dense polynomial from polynomial.
ECPSELECT : Select an extended critical pair from the extended critical pair list.
EIVFUP : Exterior integral vector from univariate integral polynomial
EX0PL : Exclude 0 from PL.
EXPPL : Exclude P from GB.
EXPTU : UGB extract exponent vector list from polynomial list.
FFFINT : Finite field element from integer.
FFGI : Floating point from gamma integer.
FFINT : Floating point from integer.
FFRN : Floating point from rational number.
FormFCond : Formula from Condition.
HDIFDI : Homogeneous distributive polynomial from distributive polynomial.
IFF : Integer from floating point.
IMFRNM : Integer matrix from rational number matrix.
IMFRNM1 : Integer matrix from rational number matrix.
IPFRP : Integral polynomial from rational polynomial.
IPSIFI : Integral polynomial standard isolating interval from isolating interval.
IVFRNV : Integer vector from rational number vector.
IVFRNV1 : Integer vector from rational number vector.
LFALL : UGB all linear forms from stack of projections 1.
LFGET : UGB get linear form from list of linear forms.
LISTS : List from string.
lvarfvlist : lvar from variable list.
MAKERN : UGB rational exponent vector list from integer ev list.
MIPFSM : Modular integral polynomial from symmetric modular.
MPPFMP : monomial of polynomial ring over polynomial ring from monomial of
NEULF : UGB compute new linear forms from new terms.
NEWDIF : UGB exponent vector list difference from polynomials.
NEWL : UGB update linear forms from new terms.
OINT : Octonion number from integer.
ORN : Octonion number from rational number.
ORNP : Octonion number from pair of rational numbers.
PFDIP : Polynomial from distributive polynomial.
PFDP : Polynomial from dense polynomial.
PFLDIPL : Recursive polynomial list (with domain-descriptor) from distributive
PLFDIL : Polynomial list from distributive polynom list.
PUFP : Polynomial, univariate, from polynomial.
PVFLISTS : permutation vector from lists.
QINT : Quaternion number from integer.
QRN : Quaternion number from rational number.
QRN4 : Quaternion number from 4-tuple of rational numbers.
RFDDFIPDD : rational function domain descriptor from integral polynomial domain

RFFIP : Rational function from integral polynomial.
RNFAP : Rational number from arbitrary precision floating point.
RNFF : Rational number from floating point.
RNINT : Rational number from integer.
RNMFIM : Rational number matrix from integer matrix.
RNVFIV : Rational number vector from integer vector.
RPFIP : Rational polynomial from integral polynomial.
SetFIntFunc : Set from integer function in domain.
SetFIPolFunc : Set from integral polynomial function in domain.
SetVlddFunc : Set variable list from domain descriptor function in domain.
SFCS : Set from characteristic set.
SLIST : String from list.
SMFMI : Symmetric modular from modular integer.
SMFMIP : Symmetric modular from modular integral polynomial.
TFDIRP : Term from distributive rational polynomial.
TFFTUPLE : type formula from coefficient tuple with joker entries.
vlistflvar : variable list from lvar.
VVECFVLIST : variable vector from variable lists.
XDIPFPF : Distributive polynomial from recursive polynomial (with domain-descriptor).
XPFDIP : Recursive polynomial (with domain-descriptor) from distributive polynomial.

full

FullRep : Full representation.

function

ADRFFADIP : arbitrary domain rational function from arbitrary domain intgeral
CallCompiled : Call compiled function or procedure.
Compiledf0 : Compiled function declaration f0.
Compiledf1 : Compiled function declaration f1.
Compiledf2 : Compiled function declaration f2.
Compiledf3 : Compiled function declaration f3.
Compiledf4 : Compiled function declaration f4.
Compiledf5 : Compiled function declaration f5.
Compiledp0 : Compiled function declaration p0.
Compiledp1 : Compiled function declaration p1.
Compiledp1v2 : Compiled function declaration p1v2.
Compiledp1v3 : Compiled function declaration p1v3.
Compiledp2 : Compiled function declaration p2.
Compiledp2v2 : Compiled function declaration p2v2.
Compiledp2v3 : Compiled function declaration p2v3.
Compiledp3 : Compiled function declaration p3.
Compiledp3v2 : Compiled function declaration p3v2.
Compiledp3v3 : Compiled function declaration p3v3.
Compiledp4 : Compiled function declaration p4.
Compiledp5 : Compiled function declaration p5.
CompSummary : Compiled function and procedure summary.
DEFE : Define expr function.
DEFF : Define fexpr function.
DEFM : Define macro function.
DEFMAP : Define generic map function.

DEFPROC : Define generic proc function.
DEFRULE : Define generic rule function.
DIIFGB : Distributive integral function polynomial groebner basis.
DIIFLS : Distributive integral function polynomial list irreducible set.
DIIFNF : Distributive integral function polynomial normal form.
DIIFSP : Distributive integral function polynom S-polynomial.
DIPRF : DIP Rational Function Definition Module.
DIPSP : DIP Integral Function Groebner Bases Implementation Module.
DIRPSR : Distributive rational polynomial symmetric function reduction.
DomLoadRF : Domain load rational function.
DOMRF : MAS Domain Rational Function Definition Module.
DSQRTF : Digit square root function.
EDIIFSUGNF : Extended distributive integral function polynomial normal with sugar
EDIIFSUGSP : Extended distributive integral function polynomial normal with sugar
ForEachinList : For each element e in r apply function f.
ForEachinRep : For each pair (n,e) in r apply function f.
IABSF : Integer absolute value function.
IFWRIT : Integral function write.
INITUPDATE : The initialization function as a first call of UPDATE.
ISIGNF : Integer sign function.
MASEXP : Exponential function.
RFDDADV : rational function domain descriptor advance.
RFDDFIPDD : rational function domain descriptor from integral polynomial domain
RFDEN : Rational function denominator.
RFDIF : Rational function difference.
RFEXP : Rational function exponentiation.
RFFIP : Rational function from integral polynomial.
RFINV : Rational function inverse.
RFNEG : Rational function negative.
RFNOV : Rational function number of variables.
RFNUM : Rational function numerator.
RFONE : Rational function one.
RFPROD : Rational function product.
RFQ : Rational function quotient.
RFREAD : Rational function read.
RFRED : Rational function reduction to lowest terms.
RFSIGN : Rational function sign.
RFSUM : Rational function sum.
RFWRIT : Rational function write.
SetAbsFunc : Set absolute value function in domain.
SetCnstFunc : Set constant test function in domain.
SetCompFunc : Set comparison function in domain.
SetConvFunc : Set conversion function in domain.
SetCPExtend : Set the critical pair extension function.
SetDdrdFunc : Set domain descriptor read function in domain.
SetDdwrFunc : Set domain descriptor write function in domain.
SetDifFunc : Set difference function in domain.
SetDIPExtend : Set the distributive polynomial extension function.
SetECPInsert : Set the extended critical pair insertion function.
SetEDIPNormalform : Set the extended distributive polynomial normalform function.
SetEDIPSPolynomial : Set the extended distributive S-polynomial function.
SetEDIPUnExtend : Set the extended distributive polynomial un-extension function.
SetExpFunc : Set exponential function in domain.

SetFactFunc : Set factorization function in domain.
SetFactoFunc : Set factorization with variable order optimization function in domain.
SetFIntFunc : Set from integer function in domain.
SetFIPolFunc : Set from integral polynomial function in domain.
SetGcdcFunc : Set gcd-and-cofactors function in domain.
SetGcdeFunc : Set gcd-and-lin-combination function in domain.
SetGcdFunc : Set gcd function in domain.
SetInvFunc : Set inversion function in domain.
SetInvTFunc : Set inversion test function in domain.
SetLcmFunc : Set lcm function in domain.
SetNegFunc : Set negation function in domain.
SetOneFunc : Set one test function in domain.
SetPCppFunc : Set Content and primitive part function.
SetPFactFunc : Set factorization function in domain.
SetPNormFunc : Set polynomial normalform function in domain.
SetProdFunc : Set product function in domain.
SetPSpolFunc : Set polynomial S-polynomial function in domain.
SetPSqfrFunc : Set polynomial squarefree factorization function in domain.
SetPSugNormFunc : Set polynomial normal with sugar strategy normalform function in domain.
SetPSugSpolFunc : Set polynomial normal with sugar strategy S-polynomial function in domain.
SetQrFunc : Set quotient and remainder function in domain.
SetQuotFunc : Set quotient function in domain.
SetReadFunc : Set read function in domain.
SetRemFunc : Set remainder function in domain.
SetSignFunc : Set sign function in domain.
SetSumFunc : Set sum function in domain.
SetToipFunc : Set conversion-to-integer-polynomial function in domain.
SetVlddFunc : Set variable list from domain descriptor function in domain.
SetWritFunc : Set write function in domain.
SEXPRP : Test if X is a S-expression function.
Signature : Signature of a compiled function or procedure.

functions

AdLoadConvFunc : arbitrary domain load conversion functions.
CGBFUNC : Comprehensive-Groebner-Bases Utility Functions Definition Module.
DIRPES : Distributive rational polynomial elementary symmetric functions.
MASELEM : MAS Elementary Functions Definition Module.
MASLOADG : MAS Load Symmetric Functions Definition Module.
SetUnion : Miscellaneous CGB Functions.
SYMMFU : Symmetric Functions Definition Module.
SYZFUNC : Syzygy Functions Definition Module.

G-Symmetric

GINBAS : G-Symmetric Integral Base Construction.
GINCHK : G-Symmetric Integral Polynomial Check.
GINCHKBAS : G-Symmetric Integral Base Check.
GINCUT : G-Symmetric Integral Polynomial Cut.
GINOPL : G-Symmetric Integral Orbit Polynomial List.
GINORP : G-Symmetric Integral Orbit Polynomial.
GINRED : G-Symmetric Integral Polynomial Reduction.
GRNBAS : G-Symmetric Rational Base Construction.
GRNCHK : G-Symmetric Rational Polynomial Check.
GRNCHKBAS : G-Symmetric Rational Base Check.
GRNCUT : G-Symmetric Rational Polynomial Cut.
GRNGGB : G-Symmetric Rational Base Construction (Buchberger-Algorithm).
GRNOPL : G-Symmetric Rational Orbit Polynomial List.
GRNORP : G-Symmetric Rational Orbit Polynomial.
GRNRED : G-Symmetric Rational Polynomial Reduction.

G-symmetric

GSDREAD : G-symmetric descriptor read.
GSPREAD : G-symmetric polynomial read.
GSRDREAD : G-symmetric rational descriptor read.
GSRREAD : G-symmetric rational polynomial read.

G-Symmetric

GSYADD : G-Symmetric Term Adder.
GSYINF : G-Symmetric Polynomial System Information.
GSYMFUIN : G-Symmetric Integral Polynomial System Definition Module.
GSYMFURN : G-Symmetric Rational Polynomial System Definition Module.
GSYMLT : G-Symmetric Multilinear Terms.
GSYNSP : G-Symmetric Number of Special Polynomials.
GSYORD : G-Symmetric Permutation Group Order.
GSYPGR : G-Symmetric Permutation Group Read.
GSYPGW : G-Symmetric Permutation Group Write.
GSYTWG : G-Symmetric Term Weight.
NOERED : Noether G-Symmetric Polynomial Computation.
SUBCHK : G-Symmetric Polynomial Check.

G-symmetric

SUBOPL : G-symmetric Orbit Polynomial List.

G-Symmetric

SUBRED : Noether G-Symmetric Polynomial Computation for Substitution Groups.

G-symmetric

SUBSYM : G-symmetric Polynomial Symmetric Check.

gamma

FFGI : Floating point from gamma integer.

gamma-integer

GREAD : Gamma-integer read.
GWRITE : Gamma-integer write.

gaussian

GDPGEN : Gaussian digit prime generator.
IMDET : Integer matrix determinant, using Gaussian elimination.
IMGE : Integer matrix Gaussian elimination.
IMGELUD : Integer matrix Gaussian elimination LU-decomposition.
RNMDET : Rational number matrix determinant, using Gaussian elimination.
RNMGE : Rational number matrix Gaussian elimination.
RNMGELUD : Rational number matrix Gaussian elimination LU-decomposition.

GB

EXPPL : Exclude P from GB.

GBTM

GBTMRED : GBTM Reduction.

GCD

DIPGCD : DIP GCD Definition Module.
MASPGCD : MAS Polynomial GCD and RES System Definition Module.
SACPGCD : SAC Polynomial GCD and RES System Definition Module.

gcd

SetGcdFunc : Set gcd function in domain.

gcd-and-cofactors

SetGcdcFunc : Set gcd-and-cofactors function in domain.

gcd-and-lin-combination

SetGcdeFunc : Set gcd-and-lin-combination function in domain.

gelfond

IPGFCB : Integral polynomial Gelfond factor coefficient bound.

general

IPGSUB : Integral polynomial general substitution.

generate

ARRAYDEC : Generate array name declarations.
EVLGIL : Exponent vector list generate for inverse lexicographical sequence.
EVLGTD : Exponent vector list generate for total degree.
GENARRAY : Generate array reference symbol.
GENINDEX : Generate index set.
GENPL : Generate parameter list.
GENPOSV : Generate Postion Vector.
GENSYM : Generate symbol.
GENSYM : Generate symbol.
GENTE : Generate typed expression.
GS1 : UGB generate stack of sorted polynomials and critical pairs 1.
GS2 : UGB generate stack of sorted polynomials and critical pairs 2.
Parse : Parse program and generate code.
TFGEN : type formula generate.
TFGENJ : type formula generate with joker argument.

generator

DPGEN : Digit prime generator.
GDPGEN : Gaussian digit prime generator.

generators

BGFUP : Base Generators Factor Update.
NLBGFUP : Non-Commutative Base Generators Factor Update.

generic

DEFMAP : Define generic map function.
DEFPROC : Define generic proc function.
DEFRULE : Define generic rule function.
SwitchParse : Switch parsing between generic / non-generic parse.

get

FORGARGS : formula get arguments.
FORGLVAR : formula get list of variables.
FORGOP : formula get operation.
FORVTGET : formula variable table get.
GET : Get property.
GET : Get property.
GetRep : Get representation.
getstck : Get contents of stack register.
gettoc : Get contents of toc register.
GREPOL : Get polynomials without green monomials.
LFGET : UGB get linear form from list of linear forms.
MGET : Matrix get.
MVFLAG : modula variable get.
MVGET : modula variable get.
pqgpol : polynomial equation get polynomial.
pqgrel : polynomial equation get relation symbol.
tfgrel : type formula get relation symbol.

given

DIRPIB : Second Algorithm for computing the involutive Base for a given F.

global

CGBGLOBRED : Comprehensive Groebner basis global reduce.
GLEXTP : Global extraneous polynomials remove.
GLOBRE : Global reduction.
MODVAR : Modula Global Variable Implementation Module.
RQEPRRC : This global variable holds information over the actual polynomial ring

graded

EVIGLC : Exponent vector inverse graded lexicographical compare.
EVRNGL : Rational exponent vector inverse graded lexicographical compare.

greatest

ADGCD : Arbitrary domain greatest common divisor.
ADGCDC : Arbitrary domain greatest common divisor and cofactors.
ADGCDE : Arbitrary domain greatest common divisor and linear combination.
AFUPGC : Algebraic number field univariate polynomial greatest common divisor
AFUPGS : Algebraic number field polynomial greatest squarefree divisor.
DEGCD : Digit extended greatest common divisor.
DGCD : Digit greatest common divisor.
EVGCD : Exponent vector greatest common divisor.
IDEGCD : Integer doubly extended greatest common divisor algorithm.
IEGCD : Integer extended greatest common divisor algorithm.
IGCD : Integer greatest common divisor.
IGCDCF : Integer greatest common divisor and cofactors.
IHEGCD : Integer half-extended greatest common divisor.
IPGCDC : Integral polynomial greatest common divisor and cofactors.
IPPGSD : Integral polynomial primitive greatest squarefree divisor.
MPGCDC : Modular polynomial greatest common divisor and cofactors.
MUPEGC : Modular univariate polynomial extended greatest common divisor.
MUPGCD : Modular univariate polynomial greatest common divisor.
MUPHEG : Modular univariate polynomial half-extended greatest common divisor.
RPBLGS : Rational polynomial base coefficients least common multiple, greatest common divisor, and sign.
RUPEGC : Rational univariate polynomial extended greatest common divisor.
RUPGCD : Rational univariate polynomial greatest common divisor.
RUPHEG : Rational univariate polynomial half-extended greatest common divisor.

green

CGBCOL : Write coloured polynomials without green monomials.
GGREEN : Write groebner-system without green monomials.
GREPOL : Get polynomials without green monomials.
MKPOL : Make polynomial without green monomials.
RMGRT : Remove green terms.

groebner

CgbCd : Groebner system initial case distinction.
CgbCons : Groebner system construct.
CGBFGSYS : Comprehensive Groebner basis from Groebner system.
CGBFGSYS : Comprehensive Groebner basis from Groebner system.
CGBGLOBRED : Comprehensive Groebner basis global reduce.
CgbI : Comprehensive Groebner basis number of conditions part.
CGBOPT : Comprehensive Groebner Basis Options.
CGBOPTWRITE : Comprehensive Groebner Basis Options Write
CgbP : Comprehensive Groebner basis polynomial list part.
CgbParts : Comprehensive Groebner basis parts.
CGBQFF : Comprehensive Groebner basis quantifier free formula.
CgbVd : Comprehensive Groebner basis variable list and domain descriptor.
CgbWrite : Comprehensive Groebner basis write.
DGBRED : Discrete Groebner Base Reduction.
DIGBC3 : Distributive polynomial groebner basis criterion 3.
DIGBC4 : Distributive polynomial groebner basis criterion 4.
DIGBMI : Distributive minimal ordered groebner basis.
DIGBZT : Distributive polynomial groebner base common zero test.
DIGMIN : Distributive minimal ordered groebner basis.
DIIFGB : Distributive integral function polynomial groebner basis.
DIIFMI : Distributive minimal ordered groebner basis.
DIIGBA : Distributive integral polynomial groebner basis augmentation.
DIIGMI : Distributive minimal ordered groebner basis.
DIIPGB : Distributive integral polynomial groebner basis.
DILFPFL : Groebner bases and related procedures for recursive integral polynomials.
DIN1GB : Distributive non-commutative polynomials Groebner base.
DINCGB : Distributive non-commutative polynomials Groebner base.
DINLGB : Distributive non-commutative polynomials left Groebner base.
DINLGM : Distributive non-commutative minimal ordered left Groebner base.
DINNGB : DIP Groebner bases for non noetherian polynomial rings.
DIPADGB : distributive polynomial arbitrary domain groebner basis.
DIPADGBext : distributive polynomial arbitrary domain groebner basis extension.
DIPADGBRED : distributive polynomial groebner basis reduction.
DIPADGBunion : distributive polynomial arbitrary domain groebner basis union.
DIPAGB : DIP Arbitrary Domain Groebner Basis Definition Module.
DIPAGB : Distributive polynomial arbitrary domain Groebner basis.
DIPDCGB : DIP Decompositional Groebner Bases Definition Module.
DIPGB : DIP Groebner Bases Definition Module.
DIPGB : Distributive polynomial groebner basis.
DIPIGB : DIP Integral Groebner Bases Definition Module.
DIPRNGB : DIP Rational Groebner Bases Definition Module.
DIPSP : DIP Integral Function Groebner Bases Implementation Module.
DIREGB : Distributive rational polynomials extended groebner basis.
DIRGBA : Distributive rational polynomial groebner basis augmentation.
DIRGBR : Distributive rational polynomial groebner basis recursion.
DIRGZS : Distributive rational Groebner base zero set.
DIRMPG : Distributive rational minimal polynomial for a groebner basis.
DIRPGB : Distributive rational polynomials groebner basis.
DNN2GB : distributive polynomials non-noetherian 2-sided Groebner base.
DNNLGB : distributive non-noetherian polynomials left Groebner base.
DNNRGB : distributive polynomials non-noetherian right Groebner base.

EVGBIT : Exponent vector groebner base intersection test.
GBE : Groebner Base with Exponent Vector Check.
GBEF : Groebner Base with Exponent Vector Check and Factors.
GBF : Groebner Base with Factors.
GBHELP : Groebner test help.
GBSYS : Groebner system.
GBSYSF : Groebner system with factorization.
GBZSET : Groebner base real zero set of zero dimensional ideal.
GroebnerBases1 : Distributive polynomials decompositional groebner bases 1.
GroebnerBases2 : Distributive polynomials decompositional groebner bases 2.
GsCd : Groebner system initial case distinction.
GsCons : Groebner system construct.
GsParts : Groebner system parts.
GsS : Groebner system system part.
GsVd : Groebner system variable list and domain descriptor.
GsWrite : Groebner system write.
GSYS : Groebner system.
GSYSDIM : Groebner system dimension.
GSYSF : Groebner system with factorization.
GSYSRED : Reduce Groebner system.
IBeqGB : Inovlutive Base equal Groebner Base.
MASNCGB : MAS Non-commutative Groebner Bases Definition Module.
MASUGB : Universal Groebner Bases Definition Module.
MGB : Modul Groebner Base.
NLDGBRED : Non-Commutative Discrete Groebner Base Reduction.
NLGBE : Non-Commutative Groebner Base with Exponent Vector Check.
NLGBEF : Non-Commutative Groebner Base with Exponent Vector Check and Factors.
NLGBF : Non-Commutative Groebner Base with Factors.
NLMGB : Non-Commutative Modul Groebner Base.
NLSPCGB : Non-Commutative S-Polynomials with Coefficients for Groebner Base.
PFIGB : Integral Polynomial Groebner Basis.
PFIGBA : Integral Polynomial Groebner Basis augmentation.
PUG : Universal Groebner base using precomputation.
PUGB : Universal Groebner base with precomputed linear forms.
SetDCGBopt : Set options for decompositional groebner bases.
SetDecompProc : Set Decomposition-Procedure for decompositional groebner bases:
SetTraceLevel : Set Trace-Level for decompositional groebner bases:
SetUpdateProc : Set Update-Procedure for decompositional groebner bases:
SetVarOrdOpt : Set Variable-Order-Optimization for decompositional groebner bases:
SPCEGB : S-Polynomials with Coefficients and Exponenetvector-Check for Groebner Base.
SPCGB : S-Polynomials with Coefficients for Groebner Base.
SYGB : Syzygy for Groebner Base.
SYGBE : Syzygy for Groebner Base with Exponent Vector.
SYZGB : Syzygy Groebner Base Definition Module.
TIPRNGB : DIP Rational Extended Groebner Bases Definition Module.
UG : Universal Groebner base.
UGB : Universal Groebner base.
WriteDCGBopt : write decompositional groebner bases options.
WRTEST : Write groebner test.
WRUGB : Write universal Groebner base.
WRUGF : Write universal Groebner family.

groebner-system

GBUPD : Groebner-system update.
GGREEN : Write groebner-system without green monomials.

groebner-System

GSRED : Groebner-System reduction.
GSYSN0 : Groebner-System n0 update.

groebner-system

WRGBS : Write groebner-system.

groebner-Test

GTEST : Groebner-Test.

groebnerBases1:

SetFacSugar : Set Factor-Sugar for procedure GroebnerBases1:

groebnerBases2:

SetBranchProc : Set Branch-Procedure for procedure GroebnerBases2:
SetReduceExp : Set Reduce-Exponent for procedure GroebnerBases2:

group

GSYORD : G-Symmetric Permutation Group Order.
GSYPGR : G-Symmetric Permutation Group Read.
GSYPGW : G-Symmetric Permutation Group Write.
GSYSPG : Symmetric Permutation Group.
SUBINF : Substitution Group Polynomial System Information.
SUBORD : Substitution Group Order.
SUBORP : Substitution Group Orbit Polynomial.
SUBSGR : Substitution Group Read.
SUBSGW : Substitution Group Write.
SUBST : Substitution Group Polynomial System Definition Module.

groups

SUBPOW : Noether SK Power Sum Computation for Substitution Groups.
SUBRED : Noether G-Symmetric Polynomial Computation for Substitution Groups.

GSYM

GSYMINP : GSYM Input Definition Module.

half-extended

IHEGCD : Integer half-extended greatest common divisor.
MUPHEG : Modular univariate polynomial half-extended greatest common divisor.
RUPHEG : Rational univariate polynomial half-extended greatest common divisor.

half-plane

IPVCHT : Integral polynomial variations after circle to half-plane transformation.
IUPCHT : Integral univariate polynomial circle to half-plane transformation.

handler

ErrorHandler : Error handler.
signal : Set system signal handler.
SigUsr1HandleDefault : SIGUSR1 default signal handler.

handling

massig : MAS Signal Handling Foreign Module.
MASSIGNAL : MAS Signal Handling Definition Module.

head

ColHT : Colouring head term.
ColpHT : Coloured polynomial head term.
ECPLCMHT : Extended critical pair select the least common multiple of head terms.

head-term

ADEPheadterm : Arbitrary domain extended plynomial head-term.

help

GBHELP : Groebner test help.
MVHLP : modula variable help.

hensel

IPIQH : Integral polynomial mod ideal quadratic Hensel lemma.
IUPQH : Integral univariate polynomial quadratic Hensel lemma.
IUPQHL : Integral univariate polynomial quadratic Hensel lemma, list.
MPIQH : Modular polynomial mod ideal, quadratic Hensel lemma.
MPIQHL : Modular polynomial mod ideal quadratic Hensel lemma, list.
MPIQHS : Modular polynomial mod ideal, quadratic Hensel lemma on a single variable.

high

IPRCH : Integral polynomial real root calculation, high precision.

high-precision

IPRCHS : Integral polynomial real root calculation, high-precision special.

higher

DIIPHD : Distributive integral polynomial higher derivation.
DIRPHD : Distributive rational polynomial higher derivation.
IPHDMV : Integral polynomial higher derivative, main variable.

highest

TESTHT : Test highest term.

hilbert

RNMHILBERT : Rational number matrix Hilbert.

holds

RQEPRRC : This global variable holds information over the actual polynomial ring

homogeneous

HDIFDI : Homogeneous distributive polynomial from distributive polynomial.
HIPRAN : Homogeneous integral polynomial random.

homogenous

HEQ : Homogenous Equation.
HSEQ : Homogenous System of Equation.
NLHEQ : Non-Commutative Homogenous Equation.
NLHSEQ : Non-Commutative Homogenous System of Equation.
SYHC : Syzygy for homogenous commutative system of equation.
SYHNL : Syzygy for homogenous non-commutative system of equation.
SYTHC : Syzygy Test for homogenous commutative Case.
SYTHNL : Syzygy Test for homogenous non-commutative Case.

homomorphism

EXMHOM : Exterior matrix homomorphism.
EXVHOM : Exterior vector homomorphism.
IPIHOM : Integral polynomial mod ideal homomorphism.
IVHOM : Integer vector homomorphism.
MAIPHM : Matrix of integral polynomials homomorphism.
MDHOM : Modular digit homomorphism.
MDVHOM : Modular vector homomorphism.
MIHOM : Modular integer homomorphism.
MIPHOM : Modular integral polynomial homomorphism.
MPHOM : Modular polynomial homomorphism.

homomorpism

FFHOM : Finite field homomorpism.

homothetic

IUPBHT : Integral univariate polynomial binary homothetic transformation.
IUPIHT : Integral univariate polynomial integer homothetic transformation.

horizontal

MTPLH : Matrix to Polynomial List Horizontal.
PLHTP : Polynomial List Horizontal To Polynomial.

i-th

DIIPEV : Distributive integral polynomial evaluation of the i-th variable.
DIPDEGI : distributive polynomial degree of i-th main variable.
DIRPEV : Distributive rational polynomial evaluation of the i-th variable.

ideal

DIPDEC0 : DIP Ideal Decomposition 0 System Definition Module.
DIPIDEAL : DIP Ideal System Definition Module.
DIPROOT : DIP Ideal Real Root System Definition Module.
DIPZ : DIP Zero Dimensional Ideal Definition Module.
DIRLCT : Distributive rational polynomial list ideal containment test.
DIRLIP : Distributive rational polynomial list ideal product.
DIRLPD : DIP rational polynomial ideal primary ideal decomposition.
DIRLPD : DIP rational polynomial ideal primary ideal decomposition.
DIRLPI : Distributive rational polynomial list primary ideal.
DIRLPW : DIP rational polynomial ideal primary ideal decomposition write.
DIRLPW : DIP rational polynomial ideal primary ideal decomposition write.
DIRPDA : DIP rational polynomial ideal primary ideal decomposition over Q(alpha).
DIRPDA : DIP rational polynomial ideal primary ideal decomposition over Q(alpha).
GBZSET : Groebner base real zero set of zero dimensional ideal.
IdealMember : ideal membership test.
InitExternalsD : Initialize external compiled ideal decomposition and root procedures.
IPIHOM : Integral polynomial mod ideal homomorphism.
IPIPR : Integral polynomial mod ideal product.
IPIQH : Integral polynomial mod ideal quadratic Hensel lemma.
MIPIPR : Modular integral polynomial mod ideal product.
MIPISE : Modular integral polynomial mod ideal, solution of equation.
MMPIQR : Modular monic polynomial mod ideal quotient and remainder.
MPIQH : Modular polynomial mod ideal, quadratic Hensel lemma.
MPIQHL : Modular polynomial mod ideal quadratic Hensel lemma, list.
MPIQHS : Modular polynomial mod ideal, quadratic Hensel lemma on a single variable.
NFEXEC : Parametric ideal membership exec.
UPDATE : Update of extended ideal basis and extended critical pair list as required
WRQFN0 : Write quantifier free formula for parametric ideal membership.

ideas

LDSSBR : Linear diophantine system solution, based on Rosser ideas.

identity

MIAIM : Matrix of integers, adjoin identity matrix, A is an m by n matrix

imaginary

CIM : Complex number imaginary part.
OIM : Octonion number imaginary part.
QIMi : Quaternion number imaginary part i.
QIMj : Quaternion number imaginary part j.
QIMk : Quaternion number imaginary part k.

implementation

DIPSP : DIP Integral Function Groebner Bases Implementation Module.
DMIA : Define model, implementation or axioms.
MLMASLOG : MAS Logic Configuration Implementation Module.
MLMLDEMO : MAS Logic Demonstration Implementation Module.
MODVAR : Modula Global Variable Implementation Module.

in-order

STLSTI : Symbol tree list, in-order.
STLSTI : Symbol tree list, in-order.

increment

IncCounter : Increment counter.

incremental

PDIF : UGB rational exponent vector list difference list, incremental.

indented

FILWRITE : Formatted indented list write.

independent

DIDIMS : Distributive polynomial dimension maximal independent set.
DIMIS : Dimension and maximal independent set.

index

DIGISM : DIP G base index search for extension multiple univariats.
DIGISR : DIP G base index search for extension reductas.
GENINDEX : Generate index set.
ILADDC : Index list addition of constant.
ILEXPR : Index list exterior product.
ILILPR : Index list inner left product.
ILINPR : Index list inner product.
ILIRPR : Index list inner right product.
ILSCMP : Index list strong compare.
ILWCMP : Index list week compare.
INDLST : Index list.
INLWRT : Index list write.
UIPSIL : Univariate integral polynomial symmetric product with exterior index list.

indexed

IXSUBS : Indexed subset.

indicators

MASERR : error indicators.
MASLISPU : Types, S-Expresion Types and Indicators.

induction

AFPRII : Algebraic number field polynomial real root isolation induction.
IBCIND : Integer binomial coefficient induction.
IPRRII : Integral polynomial real root isolation induction.

infix

FORIREAD : formula infix read.
MLDIREAD : maslog demonstration infix read.
PQIREAD : polynomial equation infix read.
TFIREAD : type formula infix read.

information

GSYINF : G-Symmetric Polynomial System Information.
NOEINF : Noether Polynomial System Information.
RQEPRRC : This global variable holds information over the actual polynomial ring
SUBINF : Substitution Group Polynomial System Information.
SysInfoStart : system information start.
SysInfoStop : system information stop.
SysInfoSum : system information sum.
SysInfoWrite : system information write.

informations

ADRMDD : arbitrary domain remove domain descriptor informations.
SYSINFO : System Informations Definition Module.

inhomogenous

IEQ : Special Solution for inhomogenous commutative equation.
ISEQ : Special Solution for inhomogenous commutative system of equation.
NLIEQ : Special Solution for inhomogenous non-commutative equation.
NLISEQ : Special Solution for inhomogenous non-commutative system of equation.
SIC : Special Solution for inhomogenous commutative system of equation.
SINL : Special Solution for inhomogenous non-commutative system of equation.
STIC : Solution Test for inhomogenous commutative Case.
STINL : Solution Test for inhomogenous non-commutative Case.

init

CDINIT : Case distinction init.
InitDIPIIB : Init distributive integral involutive base.

initial

CgbCd : Groebner system initial case distinction.
GsCd : Groebner system initial case distinction.

initialization

INITUPDATE : The initialization function as a first call of UPDATE.
SetInit : Set the initialization procedure.

initialize

initialze

inner

innermost

inovlutive

input

CGBIN : Comprehensive-Groebner-Basis input.
CLTIS : Character list to input stream.
DIBUFF : Display input buffer.
GSYMINP : GSYM Input Definition Module.
INP : Input.
IStreamKind : Input stream kind.
MLOGIO : Maslog Input Output System Definition Module.
SILINE : Set input line.
SIUNIT : Set input unit.
SUNIT1 : UGB set input unit 1.
SUNIT2 : UGB set input unit 2.
UGBBIN : UGB input, execute and output.

insert

INSPOSV : Insert Position Vector.
SetInsert : Set insert.

insertion

ECPINSERT : Extended critical pair insertion.
LEINST : List element insertion.
LINS : List insertion.
LINSRT : List insertion.
MICINS : Matrix of integers column insertion.
SetECPInsert : Set the extended critical pair insertion function.
STINS : Symbol tree insertion.
STINS : Symbol tree insertion.

intdim

INTDIM : See intdim of dip.

integer

ADFI : Arbitrary domain from integer.
ADPIQ : Arbitrary domain polynomial integer quotient.
AFFINT : Algebraic number field element from integer.
APFINT : Arbitrary precision floating point from integer.
CINT : Complex number from integer.
DIIPIP : Distributive integral polynomial integer product.
DIIPIQ : Distributive integral polynomial integer quotient.
DIPIPOL : DIP Integer Polynomial Definition Module.
DOMI : MAS Domain Integer Definition Module.
DomLoadI : Domain load integer.
DomLoadMI : Domain load modular integer.
DOMMI : MAS Domain Modular Integer Definition Module.
EIVIP : Exterior integral vector integer product.
EIVIQ : Exterior integral vector integer quotient.
FFFINT : Finite field element from integer.
FFGI : Floating point from gamma integer.
FFINT : Floating point from integer.
IABSF : Integer absolute value function.

IBCIND : Integer binomial coefficient induction.
IBCOEF : Integer binomial coefficient.
IBCPS : Integer binomial coefficient partial sum.
ICOMP : Integer comparison.
IDEGCD : Integer doubly extended greatest common divisor algorithm.
IDIF : Integer difference.
IDIPR2 : Integer digit inner product, length 2.
IDP2 : Integer division by power of 2.
IEGCD : Integer extended greatest common divisor algorithm.
IEVEN : Integer even.
IEXP : Integer exponentiation.
IFACT : Integer factorization.
IFACTL : Integer factorial.
IFCL2 : Integer, floor and ceiling, logarithm, base 2.
IFF : Integer from floating point.
IGCD : Integer greatest common divisor.
IGCDCF : Integer greatest common divisor and cofactors.
IHEGCD : Integer half-extended greatest common divisor.
IJACS : Integer Jacobi symbol algorithm.
IKM : Integer vector component product.
ILCM : Integer least common multiple.
ILCOMB : Integer linear combination.
ILOG10 : Integer logarithm base 10.
ILOG2 : Integer logarithm, base 2.
ILPDS : Integer large prime divisor search.
ILWRIT : Integer list write.
IMAX : Integer maximum.
IMDET : Integer matrix determinant, using Gaussian elimination.
IMDETL : Integer matrix determinant, using Laplace expansion.
IMDIF : Integer matrix difference.
IMFRNM : Integer matrix from rational number matrix.
IMFRNM1 : Integer matrix from rational number matrix.
IMGE : Integer matrix Gaussian elimination.
IMGELUD : Integer matrix Gaussian elimination LU-decomposition.
IMIN : Integer minimum.
IMLT : Integer lower triangular matrix transformation.
IMMAX : Integer matrix maximum norm.
IMP2 : Integer multiplication by power of 2.
IMPDS : Integer medium prime divisor search.
IMPROD : Integer matrix product.
IMSDS : Integer matrix solve decomposed system.
IMSUM : Integer matrix sum.
IMUNS : Integer matrix upper triangular matrix solution null space.
IMUT : Integer upper triangular matrix transformation.
IMWRITE : Integer matrix write.
INEG : Integer negation.
INTDDCMP : integer domain descriptor composition.
IODD : Integer odd.
IORD2 : Integer, order of 2.
IPIC : Integral polynomial integer content.
IPICPP : Integral polynomial integer content and primitive part.
IPICS : Integral polynomial integer content subroutine.
IPIP : Integral polynomial integer product.

IPIPP : Integral polynomial integer primitive part.
IPIQ : Integral polynomial integer quotient.
IPOWER : Integer power.
IPROD : Integer product.
IPROD : Integer product.
IPRODK : Integer product, Karatsuba algorithm.
IQ : Integer quotient.
IQR : Integer quotient and remainder.
IRAND : Integer, random.
IREAD : Integer read.
IREM : Integer remainder.
IROOT : Integer root.
ISEG : Integer segmentation.
ISIGNF : Integer sign function.
ISMPROD : Integer scalar and matrix product.
ISPD : Integer small prime divisors.
ISPT : Integer selfridge primality test.
ISQRT : Integer square root.
ISSUM : Integer shifted sum.
ISUM : Integer sum.
ITD : Integer trailing digit.
ITRUNC : Integer truncation.
IUM : Integer unit matrix.
IUPBEI : Integral univariate polynomial binary rational evaluation, integer output.
IUPIHT : Integral univariate polynomial integer homothetic transformation.
IVFRNV : Integer vector from rational number vector.
IVFRNV1 : Integer vector from rational number vector.
IVHOM : Integer vector homomorphism.
IVLC : Integer vector linear combination.
IVMAX : Integer vector maximum norm.
IVRAND : Integer vector random.
IVSPROD : Integer vector scalar product.
IVSQ : Integer vector scalar quotient.
IVSSUM : Integer vector scalar sum.
IVSVPROD : Integer vector scalar and vector product.
IVSVSUM : Integer vector scalar and vector sum.
IVVDIF : Integer vector difference.
IVVPROD : Integer vector vectors product.
IVVSUM : Integer vector vector sum.
IVWRITE : Integer vector write.
IWRITE : Integer write.
LINALGI : MAS Linear Algebra Integer Definition Module.
MAKERN : UGB rational exponent vector list from integer ev list.
MASI : MAS Integer Definition Module.
MASORDI : MAS order integer.
MIDCRA : Modular integer digit chinese remainder algorithm.
MIDIF : Modular integer difference.
MIEXP : Modular integer exponentiation.
MIHOM : Modular integer homomorphism.
MIINV : Modular integer inverse.
MINEG : Modular integer negation.
MIPROD : Modular integer product.

MIQ : Modular integer quotient.
MIRAN : Modular integer, random.
MISUM : Modular integer sum.
OINT : Octonion number from integer.
QINT : Quaternion number from integer.
RNINT : Rational number from integer.
RNMFIM : Rational number matrix from integer matrix.
RNMINVI : Rational number matrix inversion, integer algorithm.
RNVFIV : Rational number vector from integer vector.
SACI : SAC Integer Definition Module.
SACIPOL : SAC Integer Polynomial System Definition Module.
SACM : SAC Modular Digit and Integer Definition Module.
SetFIntFunc : Set from integer function in domain.
SKPRO2 : UGB rational exponent vector scalar product with integer ev.
SMFMI : Symmetric modular from modular integer.

integer-digit

IDPR : Integer-digit product.
IDQ : Integer-digit quotient.
IDQR : Integer-digit quotient and remainder.
IDREM : Integer-digit remainder.

integers

LBLXCO : List of beta integers lexicographical compare.
MIAIM : Matrix of integers, adjoin identity matrix, A is an m by n matrix
MICINS : Matrix of integers column insertion.
MICS : Matrix of integers column sort.
MINNCT : Matrix of integers, non-negative column transformation.
VIAZ : Vector of integers, adjoin zeros.
VIDIF : Vector of integers difference.
VIERED : Vector of integers, element reduction.
VILCOM : Vector of integers linear combination.
VINEG : Vector of integers negation.
VIPIIP : Vector of integral polynomials with vector of integers inner product.
VISPR : Vector of integers scalar product.
VISUM : Vector of integers sum.
VIUT : Vector of integers, unimodular transformation.

integral

ADFIP : Arbitrary domain from integral polynomial.
ADTOIP : Arbitrary domain to integral polynomial conversion.
AFPFIP : Algebraic number field polynomial from integral polynomial.
AFPNIP : Algebraic number field polynomial normalize to integral polynomial.
DIFIP : Distributive polynomial from distributive integral polynomial.
DIIFGB : Distributive integral function polynomial groebner basis.
DIIFLS : Distributive integral function polynomial list irreducible set.
DIIFNF : Distributive integral function polynomial normal form.
DIIFRP : Distributive integral polynomial from rational polynomial.
DIIFSP : Distributive integral function polynom S-polynomial.
DIIGBA : Distributive integral polynomial groebner basis augmentation.
DIILFR : Distributive integral polynomial list from rational polynomial list.

DIILFRCD : DIP integral list from DIP rational list using common denominator.
DIILIS : Distributive integral polynomial list irreducible set.
DIILIS : Distributive integral polynomial list irreducible set.
DIILISJ : Distributive integral polynomial list irreducible set.
DIILPP : Distributive integral polynomial list primitive part.
DIILRD : Distributive integral polynomial list read.
DIILWR : Distributive integral polynomial list write.
DIIPAB : Distributive integral polynomial absolute value.
DIIPALCMPC : Distributive integral polynomial array list check and mark polynomials.
DIIPCOM : Distributive integral polynomial complete system.
DIIPCP : Distributive integral polynomial content and primitive part.
DIIPCPLMS1 : Distributive integral polynomial list construct pairs list merge sort.
DIIPDF : Distributive integral polynomial difference.
DIIPDGB : Distributive integral polynomial D-groebner basis.
DIIPDM : Distributive integral polynomial derivation main variable.
DIIPDNF : Distributive integral polynomial normal form.
DIIPDR : Distributive integral polynomial derivation.
DIIPEGB : Distributive integral polynomial E-groebner basis.
DIIPELIMDGB : Distributive integral polynomial eliminate D-groebner base.
DIIPEM : Distributive integral polynomial evaluation of main variable.
DIIPENF : Distributive integral polynomial e-normal form.
DIIPEV : Distributive integral polynomial evaluation of the i-th variable.
DIIPEX : Distributive integral polynomial exponentiation.
DIIPGB : Distributive integral polynomial groebner basis.
DIIPGPOL : Distributive integral polynomial g polynomial.
DIIPHD : Distributive integral polynomial higher derivation.
DIIPIB : Distributive integral involutive base.
DIIPIB2 : Distributive integral polynomial involutive base.
DIIPIB3 : Distributive integral polynomial involutive base.
DIIPIP : Distributive integral polynomial integer product.
DIIPIQ : Distributive integral polynomial integer quotient.
DIIPLCPL4 : Distributive integral polynomial list construct pair list.
DIIPLEXTAL : Distributive integral polynomial list extend array list.
DIIPLM1 : Distributive integral polynomial list merge sort.
DIIPLS : Distributive integral polynomial list sum.
DIIPMN : Distributive integral polynomial maximum norm.
DIIPNF : Distributive integral polynomial normal form.
DIIPNFJ : Distributive integral polynomial normal form in the sense of Janet.
DIIPNG : Distributive integral polynomial negative.
DIIPNORM : Distributive integral polynomial norm.
DIIPON : Distributive integral polynomial one.
DIIPPR : Distributive integral polynomial product.
DIIPPR2 : Distributive integral polynomial product.
DIIPPS : Distributive integral polynomial pseudo-remainder.
DIIPQ : Distributive integral polynomial quotient.
DIIPQR : Distributive integral polynomial quotient and remainder.
DIIPRA : Distributive integral polynomial random.
DIIPRD : Distributive integral polynomial read.
DIIPREDDGB : Distributive integral polynomial reduce D-groebner base.
DIIPSG : Distributive integral polynomial sign.
DIIPSM : Distributive integral polynomial sum.
DIIPSN : Distributive integral polynomial sum norm.

DIIPSO : Distributive integral polynomial sort.
DIIPSP : Distributive integral polynomial s polynomial.
DIIPSPOL : Distributive integral polynomial s polynomial.
DIIPSPOL2 : Distributive integral polynomial s polynomial.
DIIPSU : Distributive integral polynomial substitution.
DIIPSV : Distributive integral polynomial substitution for main variable.
DIIPTDR : Distributive integral polynomial top-D-reduzibel.
DIIPTM : Distributive integral polynomial translation main variable.
DIIPTR : Distributive integral polynomial translation.
DIIPWR : Distributive integral polynomial write.
DIIPWV : Distributive integral polynomial write with standard variable list.
DIIRAS : Distributive integral polynomial random sparse exponent vector.
DILFPFL : Groebner bases and related procedures for recursive integral polynomials.
DIPEINFJ : Integral distributive extended polynomial normal form in the sense of Janet.
DIPFADIP : distributive polynomial from arbitrary domain integral polynomial.
DIPFIP : distributive polynomial from integral polynomial.
DIPI : DIP Integral Definition Module.
DIPIDGB : DIP Integral D-Groebner Bases Definition Module.
DIPIGB : DIP Integral Groebner Bases Definition Module.
DIPIIB : DIP Integral Polynomial System Definition Module in the sense of Janet.
DIPINFJ : Integral Distributive polynomial normal form in the sense of Janet.
DIPINFJS : Integral Distributive polynomial normal form in the sense of Janet modulo a
DIPSP : DIP Integral Function Groebner Bases Implementation Module.
DIRFIP : Distributive rational polynomial from integral polynomial.
DOMIP : MAS Domain Integral Polynomial Definition Module.
DomLoadIP : Domain load integral polynomials .
EDIIFSUGNF : Extended distributive integral function polynomial normal with sugar
EDIIFSUGSP : Extended distributive integral function polynomial normal with sugar
EIMWRT : Exterior integral matrix write.
EIVABS : Exterior integral vector absolute value.
EIVAPP : Exterior integral vector absolute primitive part.
EIVCPP : Exterior integral vector content and primitive part.
EIVEPR : Exterior integral vector exterior product.
EIVFUP : Exterior integral vector from univariate integral polynomial
EIVFUP : Exterior integral vector from univariate integral polynomial
EIVILP : Exterior integral vector inner left product.
EIVIP : Exterior integral vector integer product.
EIVIQ : Exterior integral vector integer quotient.
EIVIRP : Exterior integral vector inner right product.
EIVNEG : Exterior integral vector negative.
EIVPP : Exterior integral vector primitive part.
EIVSIG : Exterior integral vector sign.
EIVSUM : Exterior integral vector sum.
EIVWRT : Exterior integral vector write.
EXIDET : Exterior integral matrix determinant.
EXIDT2 : Exterior integral matrix determinant 2.
GINBAS : G-Symmetric Integral Base Construction.
GINCHK : G-Symmetric Integral Polynomial Check.
GINCHKBAS : G-Symmetric Integral Base Check.

GINCUT : G-Symmetric Integral Polynomial Cut.
GINOPL : G-Symmetric Integral Orbit Polynomial List.
GINORP : G-Symmetric Integral Orbit Polynomial.
GINRED : G-Symmetric Integral Polynomial Reduction.
GSYMFUIN : G-Symmetric Integral Polynomial System Definition Module.
HIPRAN : Homogeneous integral polynomial random.
ICHARPOL : Integral matrix characteristic polynomial.
IFWRIT : Integral function write.
IMPTRACE : Integral matrix product trace.
IMRTPROD : Integral matrix right tensor product.
IMTRACE : Integral matrix trace.
InitDIPIIB : Init distributive integral involutive base.
IPABS : Integral polynomial absolute value.
IPAFME : Integral polynomial, algebraic number field multiple evaluation.
IPC : Integral polynomial content.
IPCEVP : Integral polynomial, choice of evaluation points.
IPCPP : Integral polynomial content and primitive part.
IPCRA : Integral polynomial chinese remainder algorithm.
IPCSFB : Integral polynomial coarsest squarefree basis.
IPDDADV : integral polynomial domain descriptor advance.
IPDDCMP : integral polynomial domain descriptor composition.
IPDECMP : integral polynomial domain element composition.
IPDER : Integral polynomial derivative.
IPDIF : Integral polynomial difference.
IPDMV : Integral polynomial derivative, main variable.
IPDSCR : Integral polynomial discriminant.
IPEMV : Integral polynomial evaluation of main variable.
IPEVAL : Integral polynomial evaluation.
IPEXP : Integral polynomial exponentiation.
IPFAC : Integral polynomial factorization.
IPFCB : Integral polynomial factor coefficient bound.
IPFLC : Integral polynomial factor list combine.
IPFLMER : integral polynomial factorlist merge.
IPFRP : Integral polynomial from rational polynomial.
IPFSD : Integral polynomial factorization, second derivative.
IPFSFB : Integral polynomial finest squarefree basis.
IPGCDC : Integral polynomial greatest common divisor and cofactors.
IPGFCB : Integral polynomial Gelfond factor coefficient bound.
IPGSUB : Integral polynomial general substitution.
IPHDMV : Integral polynomial higher derivative, main variable.
IPIC : Integral polynomial integer content.
IPICPP : Integral polynomial integer content and primitive part.
IPICS : Integral polynomial integer content subroutine.
IPIHOM : Integral polynomial mod ideal homomorphism.
IPINT : Integral polynomial integration.
IPIP : Integral polynomial integer product.
IPIPP : Integral polynomial integer primitive part.
IPIPR : Integral polynomial mod ideal product.
IPIQ : Integral polynomial integer quotient.
IPIQH : Integral polynomial mod ideal quadratic Hensel lemma.
IPLCM : Integral polynomial least common multiple.
IPLCPP : Integral polynomial list of contents and primitive parts.
IPLRRI : Integral polynomial list real root isolation.

IPMAXN : Integral polynomial maximum norm.
IPNEG : Integral polynomial negative.
IPONE : Integral polynomial one.
IPPGSD : Integral polynomial primitive greatest squarefree divisor.
IPPP : Integral polynomial primitive part.
IPPROD : Integral polynomial product.
IPPSC : Integral polynomial principal subresultant coefficients.
IPPSR : Integral polynomial pseudo-remainder.
IPQ : Integral polynomial quotient.
IPQR : Integral polynomial quotient and remainder.
IPRAN : Integral polynomial random.
IPRAN : Integral polynomial, random.
IPRCH : Integral polynomial real root calculation, high precision.
IPRCHS : Integral polynomial real root calculation, high-precision special.
IPRCN1 : Integral polynomial real root calculation, 1 root.
IPRCNP : Integral polynomial real root calculation, newton method preparation.
IPREAD : Integral polynomial read.
IPRES : Integral polynomial resultant.
IPRICL : Integral polynomial real root isolation, Collins-Loos algorithm.
IPRIM : Integral polynomial real root isolation, modified Uspensky method.
IPRIMO : Integral polynomial real root isolation, modified Uspensky method, open interval.
IPRIMS : Integral polynomial real root isolation, modified Uspensky method, standard interval.
IPRIMU : Integral polynomial real root isolation, modified Uspensky method, unit interval.
IPRIU : Integral polynomial real root isolation, Uspensky method.
IPRIUP : Integral polynomial real root isolation, Uspensky method, positive roots.
IPRPRS : Integral polynomial reduced polynomial remainder sequence.
IPRRII : Integral polynomial real root isolation induction.
IPRRLS : Integral polynomial real root list separation.
IPRRRI : Integral polynomial relative real root isolation.
IPRRS : Integral polynomial real root separation.
IPSCPP : Integral polynomial sign, content, and primitive part.
IPSF : Integral polynomial squarefree factorization.
IPSFBA : Integral polynomial squarefree basis augmentation.
IPSFF : integral polynomial squarefree factorization
IPSFSD : Integral squarefree factorization, second derivative.
IPSIFI : Integral polynomial standard isolating interval from isolating interval.
IPSIGN : Integral polynomial sign.
IPSMV : Integral polynomial substitution for main variable.
IPSPRS : Integral polynomial subresultant polynomial remainder sequence.
IPSR : Integral polynomial specified roots.
IPSRM : Integral polynomial strong real root isolation, modified Uspensky method.
IPSRMS : Integral polynomial strong real root isolation, modified Uspensky method, standard interval.
IPSRP : Integral polynomial similiar to rational polynomial.
IPSUB : Integral polynomial substitution.
IPSUM : Integral polynomial sum.
IPSUMN : Integral polynomial sum norm.
IPTPR : Integral polynomial truncated product.

IPTRAN : Integral polynomial translation.
IPTRMV : Integral polynomial translation, main variable.
IPTRUN : Integral polynomial truncation.
IPVCHT : Integral polynomial variations after circle to half-plane transformation.
IPWRIT : Integral polynomial write.
ISFPF : Integral squarefree polynomial factorization.
ISFPIR : Integral squarefree polynomial isolating interval refinement.
ISIG : Integral matrix signature.
ISPSFB : Integral squarefree polynomial squarefree basis.
IUPBEI : Integral univariate polynomial binary rational evaluation, integer output.
IUPBES : Integral univariate polynomial binary rational evaluation of sign.
IUPBHT : Integral univariate polynomial binary homothetic transformation.
IUPBRE : Integral univariate polynomial binary rational evaluation.
IUPCHT : Integral univariate polynomial circle to half-plane transformation.
IUPFAC : Integral univariate polynomial factorization.
IUPFDS : Integral univariate polynomial factor degree set.
IUPIHT : Integral univariate polynomial integer homothetic transformation.
IUPMRN : Integral univariate polynomial minimal polynomial of a rational number.
IUPNT : Integral univariate polynomial negative transformation.
IUPQH : Integral univariate polynomial quadratic Hensel lemma.
IUPQHL : Integral univariate polynomial quadratic Hensel lemma, list.
IUPRB : Integral univariate polynomial root bound.
IUPRC : Integral univariate polynomial resultant and cofactor.
IUPRLP : Integral univariate polynomial, root of a linear polynomial.
IUPTPR : Integral univariate polynomial truncated product.
IUPTR : Integral univariate polynomial translation.
IUPTR1 : Integral univariate polynomial translation by 1.
IUPVAR : Integral univariate polynomial variations.
IUPVOI : Integral univariate polynomial, variations for open interval.
IUPVSI : Integral univariate polynomial, variations for standard interval.
IUSFPF : Integral univariate squarefree polynomial factorization.
MAIPDE : Matrix of integral polynomials determinant, exact division algorithm.
MAIPDM : Matrix of integral polynomials determinant, modular algorithm.
MAIPHM : Matrix of integral polynomials homomorphism.
MAIPP : Matrix of integral polynomials product.
MIPDIF : Modular integral polynomial difference.
MIPFSM : Modular integral polynomial from symmetric modular.
MIPHOM : Modular integral polynomial homomorphism.
MIPIPR : Modular integral polynomial mod ideal product.
MIPISE : Modular integral polynomial mod ideal, solution of equation.
MIPNEG : Modular integral polynomial negation.
MIPPR : Modular integral polynomial product.
MIPRAN : Modular integral polynomial, random.
MIPSUM : Modular integral polynomial sum.
MIUPQR : Modular integral univariate polynomial quotient and remainder.
MIUPSE : Modular integral univariate polynomial, solution of equation.
PFIDNOR : Integral Polynomial D Normal Form.
PFIGB : Integral Polynomial Groebner Basis.
PFIGBA : Integral Polynomial Groebner Basis augmentation.
PFILDNOR : Integral Polynomial List D-Normal Form.

PFILDS : Integral polynomial list d-irreducible set.
PFILNOR : Integral Polynomial List Normal Form.
PFILS : Integral polynomial list irreducible set.
PFINOR : Integral Polynomial Normal Form.
PFWRITE : Integral polynomial write.
RFDDFIPDD : rational function domain descriptor from integral polynomial domain
RFFIP : Rational function from integral polynomial.
RPFIP : Rational polynomial from integral polynomial.
RPMAIP : Rational polynomial monic associate of integral polynomial.
RRICOUNT : Real root integral count.
RRINFORM : Real root integral normal form.
RRINT : Real Root Integral Definition Module.
RRIPIQ : Real root integral polynomial integral quotient.
RRIPIQ : Real root integral polynomial integral quotient.
RRIPOLMATRIX : Real root integral polynomial matrix.
RRIPQSUM : Real root integral polynomial quotient sum.
RRIQUADFORM : Real root integral quadratic form.
RRISTRCONST : Real root integral structure constants.
RRIVARMATRICES : Real root integral multiplication matrices of variables.
RRUICOUNT : Real root univariate integral count.
RRUINT : Real Root Univariate Integral Definition Module.
RRUIPOLTOVEC : Real root univariate integral polynomial to vector.
RRUIQUADFORM : Real root univariate integral quadratic form.
RRUISTRCONST : Real root univariate integral structure constants.
SetDIPIIBSelect : Set Distributive integral polynomial Select.
SetFIPolFunc : Set from integral polynomial function in domain.
SMFMIP : Symmetric modular from modular integral polynomial.
UIPRES : Univariate integral polynomials resultant.
UIPRS1 : Univariate integral polynomials resultant 1.
UIPSIL : Univariate integral polynomial symmetric product with exterior index list.
UIPSIV : Univariate integral polynomial symmetric product with exterior integral vector.
UIPSIV : Univariate integral polynomial symmetric product with exterior integral vector.
VIPIIP : Vector of integral polynomials with vector of integers inner product.

integration

AFPINT : Algebraic number field polynomial integration.
IPINT : Integral polynomial integration.
RPIMV : Rational polynomial integration, main variable.

interface

InitExternalsI : Initialize external compiled interface procedures.

interpolation

MPINT : Modular polynomial interpolation.

interpreter

TfClassifyI : type formula classify coefficient tuple interpreter version.
TFGENI : TFGEN interpreter version.
TfZeroesI : TfZeroes interpreter version.

interreduce

DIPLIR : distributive polynomial list interreduce.

interreduced

DIPIRL : distributive polynomials interreduced list of polynomials.
DIPIRLJ : distributive polynomial interreduced list in the sense of Janet.
DIPIRLJ2 : Distributive polynomial list interreduced list in the sense of Janet.

intersection

CSINT : Characteristic set intersection.
EVGBIT : Exponent vector groebner base intersection test.
SINTER : Set intersection.
USINT : Unordered set intersection.

interval

ANIIPE : Algebraic number isolating interval for a primitive element.
DIITNT : Distributive polynomial system interval tupel from norm tupel.
DIITWR : Distributive polynomial system interval tupels write.
IIC : Isolating interval conversion.
IPRIMO : Integral polynomial real root isolation, modified Uspensky method, open interval.
IPRIMS : Integral polynomial real root isolation, modified Uspensky method, standard interval.
IPRIMU : Integral polynomial real root isolation, modified Uspensky method, unit interval.
IPSIFI : Integral polynomial standard isolating interval from isolating interval.
IPSIFI : Integral polynomial standard isolating interval from isolating interval.
IPSRMS : Integral polynomial strong real root isolation, modified Uspensky method, standard interval.
ISFPIR : Integral squarefree polynomial isolating interval refinement.
IUPVOI : Integral univariate polynomial, variations for open interval.
IUPVSI : Integral univariate polynomial, variations for standard interval.
RIB : Rational interval bisection.
RILC : Rational interval length comparison.
RINT : Rational interval normalizing transformation.
RIRNP : Rational interval rational number product.
RIRWRT : Rational interval refinement write.

intgeral

ADRFFADIP : arbitrary domain rational function from arbitrary domain intgeral

into

ENTER : Enter into symbol table.
ENTER : Enter into symbol table.

introduce

DILINV : distributive polynomial list introduce new variable.

introduction

DIPINV : Distributive polynomial introduction of new variables.
EVINV : Exponent vector introduction of new variables.
EVLINV : Exponent vector list introduction of new variables.
PINV : Polynomial introduction of new variables.

inverse

ADINV : Arbitrary domain inverse.
ADINVT : Arbitrary domain inverse existence test.
AFINV : Algebraic number field inverse.
CINV : Constructive inverse.
CNINV : Complex number inverse.
DILIMO : distributive polynomial list inverse monomial order.
DIPIMO : distributive polynomial inverse monomial order.
EVIGLC : Exponent vector inverse graded lexicographical compare.
EVILCI : Exponent vector inverse lexicographical compare inverse exponent vector.
EVILCI : Exponent vector inverse lexicographical compare inverse exponent vector.
EVILCP : Exponent vector inverse lexicographical compare.
EVITDC : Exponent vector inverse total degree compare.
EVLGIL : Exponent vector list generate for inverse lexicographical sequence.
EVRNGL : Rational exponent vector inverse graded lexicographical compare.
FFINV : Finite field inverse.
INV : Inverse.
INVPERM : inverse permutation.
INVPERM : inverse permutation.
MDINV : Modular digit inverse.
MIINV : Modular integer inverse.
ONINV : Octonion number inverse.
QINV : Quaternion number inverse.
RFINV : Rational function inverse.
RNINV : Rational number inverse.

inversion

RNMINV : Rational number matrix inversion.
RNMINVI : Rational number matrix inversion, integer algorithm.
SetInvFunc : Set inversion function in domain.
SetInvTFunc : Set inversion test function in domain.

involutive

DIIPIB	: Distributive integral involutive base.
DIIPIB2	: Distributive integral polynomial involutive base.
DIIPIB3	: Distributive integral polynomial involutive base.
DIPDCIB	: DIP Decompositional Involutive Bases Definition Module.
DIPIB	: Distributive polynomial involutive basis.
DIPIB2	: Distributive polynomial involutive basis.
DIPIB3	: Distributive polynom involutive base.
DIPIB4	: Distributive polynomial involutive basis.
DIRPIB	: Second Algorithm for computing the involutive Base for a given F.
DIRPIB2	: Distributive rational polynom involutive basis.
IBLWR	: Involutive bases list write.
InitDIPIIB	: Init distributive integral involutive base.
InvolutiveBases	: Involutive Bases.
IsInvolutive	: Is involutive.
SetDCIBDecomp	: Set decompositional involutive base decomposition.
SetDCIBdepth	: Set decompositional involutive base depth of tree.
SetDCIBopt	: Set decompositional involutive base options.
SetDCIBTraceLevel	: Set Decompositional involutive base Trace Level.
SetDCIBVarOrdOpt	: Set decompositional involutive base variable order option.
SetDIPIBCancel	: Set distributive polynomial involutive base cancel.
SetDIPIBCrit	: Set distributive polynomial involutive base critera.
SetDIPIBISJ	: Set distributive involutive base irreducible set in the sense of Janet.
SetDIPIBopt	: Set distributive polynomial involutive base options.
SetDIPIBSelect	: Set distributive polynomial involutive base select.

IO

Summary	: Summary of stream IO.

irreducible

DIIFLS	: Distributive integral function polynomial list irreducible set.
DIILIS	: Distributive integral polynomial list irreducible set.
DIILIS	: Distributive integral polynomial list irreducible set.
DIILISJ	: Distributive integral polynomial list irreducible set.
DILIS	: Distributive polynomial list irreducible set.
DILISJ	: Distributive polynomial list irreducible set in the sense of Janet.
DILISJ2	: Distributive polynomial list irreducible set.
DINLIS	: Distributive non-commutative polynomial list left irreducible set.
DIPADIRSET	: distributive polynomial arbitrary domain irreducible set.
DIRLIS	: Distributive rational polynomial list irreducible set.
DIRLISJ	: Distributive rational polynomial list irreducible set.
PFILS	: Integral polynomial list irreducible set.
SetDIPIBISJ	: Set distributive involutive base irreducible set in the sense of Janet.

is

IsInvolutive	: Is involutive.

isolating

ANIIPE : Algebraic number isolating interval for a primitive element.
IIC : Isolating interval conversion.
IPSIFI : Integral polynomial standard isolating interval from isolating interval.
IPSIFI : Integral polynomial standard isolating interval from isolating interval.
ISFPIR : Integral squarefree polynomial isolating interval refinement.

isolation

AFPBRI : Algebraic number field polynomial basis real root isolation.
AFPCLL : Algebraic number field polynomial real root isolation, Collins-Loos
AFPRCL : Algebraic number field polynomial real root isolation, Collins-Loos algorithm.
AFPRII : Algebraic number field polynomial real root isolation induction.
AFPRRI : Algebraic number field polynomial relative real root isolation.
IPLRRI : Integral polynomial list real root isolation.
IPRICL : Integral polynomial real root isolation, Collins-Loos algorithm.
IPRIM : Integral polynomial real root isolation, modified Uspensky method.
IPRIMO : Integral polynomial real root isolation, modified Uspensky method, open interval.
IPRIMS : Integral polynomial real root isolation, modified Uspensky method, standard interval.
IPRIMU : Integral polynomial real root isolation, modified Uspensky method, unit interval.
IPRIU : Integral polynomial real root isolation, Uspensky method.
IPRIUP : Integral polynomial real root isolation, Uspensky method, positive roots.
IPRRII : Integral polynomial real root isolation induction.
IPRRRI : Integral polynomial relative real root isolation.
IPSRM : Integral polynomial strong real root isolation, modified Uspensky method.
IPSRMS : Integral polynomial strong real root isolation, modified Uspensky method, standard interval.

jacobi

IJACS : Integer Jacobi symbol algorithm.

janet

ADEPNFJ : Arbitrary domain extended polynomial normalform in the sense of Janet.
ADPNFJ : Arbitrary domain polynomial normalform in the sense of Janet.
ADPNFJS : Arbitrary domain polynomial normalform in the sense of Janet modulo a set
DIIPNFJ : Distributive integral polynomial normal form in the sense of Janet.
DILISJ : Distributive polynomial list irreducible set in the sense of Janet.
DILNFJ : Distributive Polynomial List normalform in the sense of Janet.
DIPCJ : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPEINFJ : Integral distributive extended polynomial normal form in the sense of Janet.
DIPENFJ : Distributive extended polynomial normal form in the sense of Janet.

DIPIB : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPIIB : DIP Integral Polynomial System Definition Module in the sense of Janet.
DIPINFJ : Integral Distributive polynomial normal form in the sense of Janet.
DIPINFJS : Integral Distributive polynomial normal form in the sense of Janet modulo a
DIPIRLJ : distributive polynomial interreduced list in the sense of Janet.
DIPIRLJ2 : Distributive polynomial list interreduced list in the sense of Janet.
DIPNFJ : Distributive polynomial normal form in the sense of Janet.
DIPNFJS : Distributive polynomial normal form in the sense of Janet modulo a set of
DIPRNIB : DIP Rational Numbers Polynomial Definition Module in the sense of Janet.
DIRPNFJ : Distributive rational polynomial normal form in the sense of Janet.
EVMTJ : Exponent vector multiple test in the sense of Janet.
SetDIPIBISJ : Set distributive involutive base irreducible set in the sense of Janet.

janet-reducible

ADGJredI : Arbitrary domain polynomial G Janet-reducible modulo I.

jdomain

SetDomainNFJ : Initialize Jdomain.

joker

TfCtj : type formula coefficient tuples with joker argument.
TFFTUPLE : type formula from coefficient tuple with joker entries.
TFGENJ : type formula generate with joker argument.

jump

longjmp : Long jump to old environment.
setjmp : Set jump environment.

kannan

LDSMKB : Linear diophantine system solution, modified Kannan and Bachem algorithm.

karatsuba

IPRODK : Integer product, Karatsuba algorithm.

key

KEYCOL : Key colour.
KEYREAD : key read.
SEENR : Find key for option.

kind

EStreamKind : Error stream kind.
IStreamKind : Input stream kind.
OStreamKind : Output stream kind.

kpathsearch

maskpathsea : kpathsearch Foreign Module.

kreisteilungs

KREISP : Kreisteilungs polynom.

lambda

LAMBDAP : Test if expression S is a lambda form.

laplace

IMDETL : Integer matrix determinant, using Laplace expansion.
RNMDETL : Rational number matrix determinant, using Laplace expansion.

large

ILPDS : Integer large prime divisor search.

last

ADDLAST : Add last Polynomial.
DO1 : UGB add last component to exponent vector.
LAST : Last.
LASTEL : Last element.

lcm

SetLcmFunc : Set lcm function in domain.

leading

ADEPleadingterm : Arbitrary polynomial leading term.
APNELD : Arbitrary precision floating point number of equal leading digits.
DIPEVL : Distributive polynomial exponent vector leading monomial.
DIPLBC : Distributive polynomial leading base coefficient.
DIPLDC : Distributive polynomial leading coefficient.
EDIPEVL : Extended distributive polynomial exponent vector of the leading monomial.
PLBCF : Polynomial leading base coefficient.
PLDCF : Polynomial leading coefficient.

least

ADLCM : Arbitrary domain least common multiple.
ECPLCMHT : Extended critical pair select the least common multiple of head terms.
EVLCM : Exponent vector least common multiple.
ILCM : Integer least common multiple.
IPLCM : Integral polynomial least common multiple.
RPBLGS : Rational polynomial base coefficients least common multiple, greatest common divisor, and sign.
RUPLCM : Rational univariate polynomial least common multiple.

left

DINLGB : Distributive non-commutative polynomials left Groebner base.
DINLGM : Distributive non-commutative minimal ordered left Groebner base.
DINLIS : Distributive non-commutative polynomial list left irreducible set.
DINLMPG : Distributive non-commutative left rational minimal polynomial for a G basis.
DINLMPL : Distributive non-commutative left rational minimal polynomial list for a G basis.
DINLNF : Distributive non-commutative polynomial left normal form.
DINLSP : Distributive non-commutative polynomial left S-polynomial.
DNLCPL : distributive polynomial non-noetherian left construct pair list.
DNLUPL : distributive polynomial non-noetherian left update pair list.
DNNLGB : distributive non-noetherian polynomials left Groebner base.
EIVILP : Exterior integral vector inner left product.
ILILPR : Index list inner left product.

lemma

IPIQH : Integral polynomial mod ideal quadratic Hensel lemma.
IUPQH : Integral univariate polynomial quadratic Hensel lemma.
IUPQHL : Integral univariate polynomial quadratic Hensel lemma, list.
MPIQH : Modular polynomial mod ideal, quadratic Hensel lemma.
MPIQHL : Modular polynomial mod ideal quadratic Hensel lemma, list.
MPIQHS : Modular polynomial mod ideal, quadratic Hensel lemma on a single variable.

length

EVL : Exponent Vector Length.
IDIPR2 : Integer digit inner product, length 2.
LENGTH : Length.
RILC : Rational interval length comparison.

letter

LETTER : Letter.

level

DLSWRITE : debug level SWRITE.
POLCOP : Two level list copy.
SetDCIBTraceLevel : Set Decompositional involutive base Trace Level.
SetDIPIBTraceLevel : Set Trace Level.

lex

DINPTslT : Distributive polynomial non-commutative product table strict lex test.

lexicographical

EVIGLC : Exponent vector inverse graded lexicographical compare.
EVILCI : Exponent vector inverse lexicographical compare inverse exponent vector.
EVILCP : Exponent vector inverse lexicographical compare.
EVLGIL : Exponent vector list generate for inverse lexicographical sequence.
EVRNGL : Rational exponent vector inverse graded lexicographical compare.
LBLXCO : List of beta integers lexicographical compare.

lexicographically

CPLEXN : Cartesian product, lexicographically next.
LEXNEX : Lexicographically next.

like

MWRIT1 : Output in modula like syntax.
MWRITE : Output in modula like syntax.

line

masReadL : MAS Read Line.
SILINE : Set input line.
SOLINE : Set output line.

linear

ADGCDE : Arbitrary domain greatest common divisor and linear combination.
AFUPRL : Algebraic number field polynomial, root of a linear polynomial.
ALFA : Automatic Linear Form Adaption.
ALFRA : Automatic Linear Form Readaption.
ALLELN : UGB all linear forms from stack of projections.
ALLLF : UGB all linear forms from stack of projections and print.
CLF2 : UGB compute linear form from difference set 2.
CLF3 : UGB compute linear form from difference set 3.
COMPLF : UGB compute linear form from difference set.
CP2 : UGB linear form product with rational exponent vector list 2.
CQ2 : UGB linear form product with rational exponent vector list.
CSPUR : UGB trace for linear form 2.
EVLFCP : Exponent vector linear form compare.
EVLFCP : UGB exponent vector linear form compare.
ILCOMB : Integer linear combination.
InitExternalsL : Initialize external compiled linear algebra procedures.
ISNEUL : UGB new linear form test.
IUPRLP : Integral univariate polynomial, root of a linear polynomial.
IVLC : Integer vector linear combination.
LDSMKB : Linear diophantine system solution, modified Kannan and Bachem algorithm.
LDSSBR : Linear diophantine system solution, based on Rosser ideas.
LF : UGB linear form.

LFALL : UGB all linear forms from stack of projections 1.
LFCHECK : Linear form check.
LFGET : UGB get linear form from list of linear forms.
LFGET : UGB get linear form from list of linear forms.
LINALG : Linear algebra definition module.
LINALGI : MAS Linear Algebra Integer Definition Module.
LINALGRN : MAS Linear Algebra Rational Number Definition Module.
MKLF1 : UGB make new linear forms 1.
MKLF2 : UGB make new linear forms 2.
MKLF3 : UGB make new linear forms 3.
NEULF : UGB compute new linear forms from new terms.
NEWL : UGB update linear forms from new terms.
NONEWL : UGB update linear forms without new terms.
PKEGEL : UGB trace for linear form.
PLF : UGB linear form with precomputed linear forms.
PLF : UGB linear form with precomputed linear forms.
PUGB : Universal Groebner base with precomputed linear forms.
RNVLC : Rational number vector linear combination.
SACLDIO : SAC Linear Diophantine Equation System Definition Module.
VILCOM : Vector of integers linear combination.
ZULFO : UGB find admissible extensions of linear forms.

lines

BLINES : Blank lines.

lisp

EVALUATE : Lisp evaluator.

LISP

InitExternalsG : Tell Modula and LISP about external compiled procedures.
InitExternalsS : Tell Modula and LISP about external compiled procedures.

lisp

MASLISP : MAS Lisp Definition Module.
MASLISPU : MAS Lisp Utility Definition Module.

list

ADDDFDIL : arbitrary domain domain descriptor from distributive polynomial list.
ADDDFDILD : arbitrary domain domain descriptor from distributive polynomial list
ADDNFDIL : arbitrary domain domain number from distributive polynomial list.
ADDNFDILD : arbitrary domain domain number from distributive polynomial list
ADLGeqH : Arbitrary domain polynomial list G equal H.
ADLGinH : Arbitrary domain polynomial list G in H.
ADVLDD : variable list from domain descriptor.
AFPRLS : Algebraic number field polynomial real root list separation.
CdpVd : Case distinction and polynomial set variable list and domain descriptor
CGBLM : CGB coloured distributive polynomial list merge.
CGBLPM : CGB list merge.

CgbP : Comprehensive Groebner basis polynomial list part.
CgbVd : Comprehensive Groebner basis variable list and domain descriptor.
CHDEGL : Check degree of polynomial list.
CLIN : Character list in.
CLISTFA : character list from atom.
CLOUT : Character list out.
CLTIS : Character list to input stream.
COMPA1 : UGB trace member in trace list.
CP2 : UGB linear form product with rational exponent vector list 2.
CQ2 : UGB linear form product with rational exponent vector list.
DCLWR : Coloured polynomials list write.
DEGRE : UGB total degree of a list of rational exponent vectors.
DET : Determine list of polynomials.
DIDPALCMPC : Distributive domain polynomial array list check and mark polynomials.
DIDPCPLMS1 : Distributive domain polynomial list construct pairs list merge sort.
DIDPCPLMS1 : Distributive domain polynomial list construct pairs list merge sort.
DIDPLCPL4 : Distributive domain polynomial list construct pair list.
DIDPLCPL4 : Distributive domain polynomial list construct pair list.
DIDPLEXTAL : Distributive domain polynomial list extend array list.
DIDPLEXTAL : Distributive domain polynomial list extend array list.
DIDPLM1 : Distributive domain polynomial list merge sort.
DIDPUCPL1 : Distributive domain polynomial update constructed pairs list.
DIFF : UGB difference set for rational exponent vector list.
DIFF1 : UGB difference set for two rational exponent vector list.
DIIFLS : Distributive integral function polynomial list irreducible set.
DIILFR : Distributive integral polynomial list from rational polynomial list.
DIILFR : Distributive integral polynomial list from rational polynomial list.
DIILFRCD : DIP integral list from DIP rational list using common denominator.
DIILFRCD : DIP integral list from DIP rational list using common denominator.
DIILIS : Distributive integral polynomial list irreducible set.
DIILIS : Distributive integral polynomial list irreducible set.
DIILISJ : Distributive integral polynomial list irreducible set.
DIILPP : Distributive integral polynomial list primitive part.
DIILRD : Distributive integral polynomial list read.
DIILWR : Distributive integral polynomial list write.
DIIPALCMPC : Distributive integral polynomial array list check and mark polynomials.
DIIPCPLMS1 : Distributive integral polynomial list construct pairs list merge sort.
DIIPCPLMS1 : Distributive integral polynomial list construct pairs list merge sort.
DIIPLCPL4 : Distributive integral polynomial list construct pair list.
DIIPLCPL4 : Distributive integral polynomial list construct pair list.
DIIPLEXTAL : Distributive integral polynomial list extend array list.
DIIPLEXTAL : Distributive integral polynomial list extend array list.
DIIPLM1 : Distributive integral polynomial list merge sort.
DIIPLS : Distributive integral polynomial list sum.
DIIPUCPL1 : Distributive polynomial D-update constructed pairs list.
DIIPWV : Distributive integral polynomial write with standard variable list.
DILADNF : distributive polynomial list arbitrary domain normal form.
DILATDG : Distributive polynom list add total degree.
DILBBS : Distributive List Bubble Sort.
DILBSO : Distributive polynomial list bubble sort.
DILCONV : distributive polynomial list conversion.

DILCPL : Distributive polynomial list construct pair list.
DILCPL : Distributive polynomial list construct pair list.
DILDIM : Distributive polynomial list dimension.
DILEBBS : Distributive List Extended Bubble Sort.
DILEP2P : Distributive polynom list extended polynom to polynom.
DILFDILP : distributive polynomial list from distributive polynomial list over
DILFDILP : distributive polynomial list from distributive polynomial list over
DILFEL : Distributive polynomial list from exponent vector list.
DILFEL : Distributive polynomial list from exponent vector list.
DILFPFL : Distributive polynomial list with arbitrary domain coefficients from
DILFPL : Distributive polynomial list from polynom list.
DILFPL : Distributive polynomial list from polynom list.
DILIMO : distributive polynomial list inverse monomial order.
DILINV : distributive polynomial list introduce new variable.
DILIS : Distributive polynomial list irreducible set.
DILISJ : Distributive polynomial list irreducible set in the sense of Janet.
DILISJ2 : Distributive polynomial list irreducible set.
DILNFJ : Distributive Polynomial List normalform in the sense of Janet.
DILPERM : distributive polynomial list permutation of variables.
DILPFDIL : distributive polynomials over polynomial ring list from distributive
DILPROD : distributive polynomial list product.
DILRD : Distributive polynomial list read.
DILSUM : Distributive polynomial list sum.
DILTDG : Distributive polynomial list total degree
DILUPL : Distributive polynomial list update pair list.
DILUPL : Distributive polynomial list update pair list.
DILWR : Distributive polynomial list write.
DINLIS : Distributive non-commutative polynomial list left irreducible set.
DINLMPL : Distributive non-commutative left rational minimal polynomial list for a G basis.
DINLRD : Distributive non-commutative polynomial list read.
DIPIRL : distributive polynomials interreduced list of polynomials.
DIPIRLJ : distributive polynomial interreduced list in the sense of Janet.
DIPIRLJ2 : Distributive polynomial list interreduced list in the sense of Janet.
DIPIRLJ2 : Distributive polynomial list interreduced list in the sense of Janet.
DIPLDM : Distributive polynomial list degree matrix.
DIPLDV : Distributive polynomial list dependency on variables.
DIPLFPFL : Distributive polynomial list from recursive polynomial
DIPLIR : distributive polynomial list interreduce.
DIPLM : Distributive polynomial list merge.
DIPLMD : distributive polynomial list maximum degree.
DIPLPM : Distributive polynomial list pair-merge sort.
DIPLRS : Distributive polynomial list re-sort.
DIPNML : Distributive polynomial nonmultiple variable list.
DIPPGL : Distributive polynomial prolongation list.
DIPPGL2 : Distributive polynomial prolongation list.
DIPPGL3 : Distributive polynom prolongation list.
DIPRLF : distributive polynomials reduce list of polynomials with factor.
DIPVDEF : DIP define distributive polynomial variable list.
DIPVL : Distributive Polynomial List of Variables.
DIRLCT : Distributive rational polynomial list ideal containment test.
DIRLIP : Distributive rational polynomial list ideal product.
DIRLIS : Distributive rational polynomial list irreducible set.

DIRLISJ : Distributive rational polynomial list irreducible set.
DIRLPI : Distributive rational polynomial list primary ideal.
DIRLRD : Distributive rational polynomial list read.
DIRLWR : Distributive rational polynomial list write.
DIRPLS : Distributive rational polynomial list sum.
DNLCPL : distributive polynomial non-noetherian left construct pair list.
DNLUPL : distributive polynomial non-noetherian left update pair list.
DNRCPL : distributive polynomial non-noetherian right construct pair list.
DNRUPL : distributive polynomial non-noetherian right update pair list.

LIST

dummyfactorize : LIST output.

list

ECPSELECT : Select an extended critical pair from the extended critical pair list.
EVLGIL : Exponent vector list generate for inverse lexicographical sequence.
EVLGTD : Exponent vector list generate for total degree.
EVLINV : Exponent vector list introduction of new variables.
EVLRNBSO : Rational exponent vector list bubble sort.
EXPTU : UGB extract exponent vector list from polynomial list.
EXPTU : UGB extract exponent vector list from polynomial list.
FdV : Formula and dimension variable list part.
FILWRITE : Formatted indented list write.
FLWRITE : Formatted list write.
FORGLVAR : formula get list of variables.
FORISLVAR : formula is variable list.
FORMKLVAR : formula make list of variables.
FORPLVAR : formula parse list of variables.
FORRDLVAR : formula read list of variables.
FORTEXWLVAR : tex write list of varaiables.
FRESL : Fermat residue list.
FRLSM : Fermat residue list, single modulus.
GENPL : Generate parameter list.
GINOPL : G-Symmetric Integral Orbit Polynomial List.
GRNOPL : G-Symmetric Rational Orbit Polynomial List.
GsVd : Groebner system variable list and domain descriptor.
IBLWR : Involutive bases list write.
ILADDC : Index list addition of constant.
ILEXPR : Index list exterior product.
ILILPR : Index list inner left product.
ILINPR : Index list inner product.
ILIRPR : Index list inner right product.
ILSCMP : Index list strong compare.
ILWCMP : Index list week compare.
ILWRIT : Integer list write.
INDLST : Index list.
INLWRT : Index list write.
IPFLC : Integral polynomial factor list combine.
IPLCPP : Integral polynomial list of contents and primitive parts.
IPLRRI : Integral polynomial list real root isolation.
IPRRLS : Integral polynomial real root list separation.
IUPQHL : Integral univariate polynomial quadratic Hensel lemma, list.

LBIBMS : List of beta-integers bubble-merge sort.
LBIBS : List of beta-integers bubble sort.
LBIM : List of beta-integers merge.
LBLXCO : List of beta integers lexicographical compare.
LCONC : List concatenation.
LDEG : Distributive polynomial list total degree.
LDIPEXTEND : List of distributive polynomials extend.
LECPUNEXTEND : List of extended critical pairs un-extend.
LECPWRITE : List of extended critical pairs write.
LEDIPUNEXTEND : List of extended distributive polynomials un-extend.
LEDIPWRITE : List of extended distributive polynomials write.
LEINST : List element insertion.
LELT : List element.
LEQUAL : List equality.
LEROT : List element rotation.
LFGET : UGB get linear form from list of linear forms.
LINS : List insertion.
LINSRT : List insertion.
LIST1 : List, 1 element.
LIST10 : List, 10 elements.
LIST2 : List, 2 elements.
LIST3 : List, 3 elements.
LIST4 : List, 4 elements.
LIST5 : List, 5 elements.
LIST6 : list of 6 elements.
LISTS : List from string.
LISTTOOLS : List Tools Definition Module.
LISTVAR : List variable.
LMERGE : List merge.
LPAIRS : list pairs.
LPERM : List permute.
LREAD : List read.
LRNBMS : List of rational numbers bubble-merge sort.
LRNBS : List of rational numbers bubble sort.
LRNM : List of rational numbers merge.
LRNWRIT : List of rational numbers write.
LSRCH : List search.
LSRCHQ : List search equal.
lvarfvlist : lvar from variable list.
LWRITE : List write.
MAKERN : UGB rational exponent vector list from integer ev list.
MAKERN : UGB rational exponent vector list from integer ev list.
MCOEF : Make coefficient list.
MDLCRA : Modular digit list chinese remainder algorithm.
MINPP : Minimize polynomials list.
MKPAIR : UGB make critical pairs for polynomial list.
MKSET : UGB rational exponent vector list difference list.
MKSET : UGB rational exponent vector list difference list.
MPIQHL : Modular polynomial mod ideal quadratic Hensel lemma, list.
MTPLH : Matrix to Polynomial List Horizontal.
MTPLV : Matrix to Polynomial List Vertical.
MVDeclareL : modula variable declare list.
NEWDIF : UGB exponent vector list difference from polynomials.

NLPLMULT : Non-Commutative Polynomial List Multiplication.
PACK : Pack character list.
PACK : Pack character list.
PBCLI : Polynomial base coefficients list.
PCL : Polynomial coefficient list.
PdF : Parametric dimension formula and dimension list part.
PDIF : UGB rational exponent vector list difference list, incremental.
PDIF : UGB rational exponent vector list difference list, incremental.
PdVd : Parametric dimension variable list and domain descriptor part.
PFILDNOR : Integral Polynomial List D-Normal Form.
PFILDS : Integral polynomial list d-irreducible set.
PFILNOR : Integral Polynomial List Normal Form.
PFILS : Integral polynomial list irreducible set.
PFLDIPL : Recursive polynomial list (with domain-descriptor) from distributive
PLFDIL : Polynomial list from distributive polynom list.
PLFDIL : Polynomial list from distributive polynom list.
PLHTP : Polynomial List Horizontal To Polynomial.
PLMULT : Polynomial List Multiplication.
PLVTM : Polynomial List Vertical To Matrix.
PLWR : Polynomial List Write.
POLCOP : Two level list copy.
RNVABS : Rational number list absolute values.
RPLWRS : Rational polynomial list write.
SACLIST : SAC List Processing Definition Module.
SetDIPAGBOptions : Set the trace flag, the strategy and the variable weight list of the
SetDIPAGBVariableWeight : Set the DIPAGB variable weight list for the normal with sugar strategy.
SetVlddFunc : Set variable list from domain descriptor function in domain.
SLELT : Set list element.
SLIST : String from list.
STLST : Symbol tree list.
STLST : Symbol tree list.
STLSTI : Symbol tree list, in-order.
STLSTI : Symbol tree list, in-order.
STNLST : Symbol tree nodes list.
STVL : Standard variable list.
SUBLIS : Substitution with list.
SUBLIS : Substitution with list.
SUBOPL : G-symmetric Orbit Polynomial List.
TCOMP : UGB list constructive conc.
UIPSIL : Univariate integral polynomial symmetric product with exterior index list.
UPCASE : upcase character list.
UPDATE : Update of extended ideal basis and extended critical pair list as required
UpdateVariableWeight : Update of the DIPAGB variable weight list.
VdCons : Variable list and domain descriptor construct.
VdD : Variable list and domain descriptor domain descriptor part.
VdParts : Variable list and domain descriptor parts.
VdRead : Variable list and domain descriptor read.
VdV : Variable list and domain descriptor variable list part.
VdV : Variable list and domain descriptor variable list part.
vlistflvar : variable list from lvar.

VLREAD : Variable list read.
VLSRCH : Variable list search.
VLWRIT : Variable list write.
WriteDIPAGBOptions : Write the current trace flag, strategy and variable weight list of the
WriteDIPAGBVariableWeight : Write the DIPAGB variable weight list in the output stream.

LIST;

DIPDCGB : TYPE PROCF1 = PROCEDURE(LIST): LIST;

lists

EQPLCL : Equal lists of coloured polynomials.
PVFLISTS : permutation vector from lists.
VVECFVLIST : variable vector from variable lists.

ln

LN : Ln.

load

AdLoadConvFunc : arbitrary domain load conversion functions.
DomLoadAF : Domain load algebraic number.
DomLoadAPF : Domain load arbitrary precision floating point.
DomLoadC : Domain load complex number.
DomLoadFF : Domain load finite field.
DomLoadI : Domain load integer.
DomLoadIP : Domain load integral polynomials .
DomLoadMD : Domain load modular digit.
DomLoadMI : Domain load modular integer.
DomLoadO : Domain load octonion number.
DomLoadQ : Domain load quaternion number.
DomLoadRF : Domain load rational function.
DomLoadRN : Domain load rational number.
DomLoadRP : Domain load rational polynomials .
MASLOAD : MAS Load Definition Module.
MASLOADA : MAS Load Definition Module A.
MASLOADB : MAS Load Definition Module B.
MASLOADC : MAS Load Definition Module C.
MASLOADD : MAS Load Definition Module D.
MASLOADE : MAS Load Definition Module E.
MASLOADG : MAS Load Symmetric Functions Definition Module.
MASLOADJ : MAS Load Definition Module J.
MASLOADL : MAS Load Definition Module L.
MASLOADM : MAS Load Definition Module M.
MASLOADQ : MAS Load Definition Module Q.
MASLOADS : MAS Load Syzygy Definition Module.

log

LOG : Log.

logarithm

APLG10 : Arbitrary precision floating point logarithm base 10.
DLOG2 : Digit logarithm, base 2.
FLOG10 : Floating point logarithm base 10.
IFCL2 : Integer, floor and ceiling, logarithm, base 2.
ILOG10 : Integer logarithm base 10.
ILOG2 : Integer logarithm, base 2.
RNFCL2 : Rational number floor and ceiling of logarithm, base 2.

logic

InitExternalsML : Initialze external compiled logic procedures.
MLMASLOG : MAS Logic Configuration Implementation Module.
MLMLDEMO : MAS Logic Demonstration Implementation Module.

long

longjmp : Long jump to old environment.

lookup

DINPTL : Distributive polynomial non-commutative product table lookup.

lower

IMLT : Integer lower triangular matrix transformation.
RNMLT : Rational matrix lower triangular matrix transformation.

lowest

RFRED : Rational function reduction to lowest terms.
RNRED : Rational number reduction to lowest terms.

LU-decomposition

IMGELUD : Integer matrix Gaussian elimination LU-decomposition.
RNMGELUD : Rational number matrix Gaussian elimination LU-decomposition.

lvar

FORPPLVAR : formula print lvar.
lvarfvlist : lvar from variable list.
vlistflvar : variable list from lvar.

macro

DEFM : Define macro function.

main

AFPDMV : Algebraic number field polynomial derivative, main variable.
AFPEMV : Algebraic number field polynomial evaluation of main variable.
CGBMAIN : Comprehensive-Groebner-Bases Main Programms Definition Module.
DIIPDM : Distributive integral polynomial derivation main variable.
DIIPEM : Distributive integral polynomial evaluation of main variable.
DIIPSV : Distributive integral polynomial substitution for main variable.
DIIPTM : Distributive integral polynomial translation main variable.
DIPADM : Distributive polynomial advance main variable.
DIPDEGI : distributive polynomial degree of i-th main variable.
DIPMPM : Distributive polynomial multiplication by power of main variable.
DIRPDM : Distributive rational polynomial derivation main variable.
DIRPEM : Distributive rational polynomial evaluation of main variable.
DIRPSV : Distributive rational polynomial substitution for main variable.
DIRPTM : Distributive rational polynomial translation main variable.
IPDMV : Integral polynomial derivative, main variable.
IPEMV : Integral polynomial evaluation of main variable.
IPHDMV : Integral polynomial higher derivative, main variable.
IPSMV : Integral polynomial substitution for main variable.
IPTRMV : Integral polynomial translation, main variable.
MCPMV : Matrix of coefficients of polynomials, with respect to main variable.
MPEMV : Modular polynomial evaluation of main variable.
PMPMV : Polynomial multiplication by power of main variable.
RPDMV : Rational polynomial derivative, main variable.
RPEMV : Rational polynomial evaluation, main variable.
RPIMV : Rational polynomial integration, main variable.
SYZMAIN : Syzygy Main Programs Definition Module.

make

FORMKBINOP : formula make binary operation.
FORMKCNF : formula make cnf.
FORMKCNST : formula make constant, i.
FORMKDNF : formula make dnf.
FORMKFOR : formula make formula.
FORMKLVAR : formula make list of variables.
FORMKPOS : formula make positive.
FORMKPRENEX : formula make prenex.
FORMKPRENEX1 : formula make prenex 1.
FORMKPRENEXI : formula make prenex.
FORMKQUANT : formula make quantifier.
FORMKUNOP : formula make unary operation.
FORMKVAR : formula make variable.
FORMKVD : formula make variable names disjoint.
MCOEF : Make coefficient list.
MKACOL : Make colour.
MKCGB : Make Comprehensive-Groebner-Basis.
MKCOL : Make new colour.
MKLF1 : UGB make new linear forms 1.
MKLF2 : UGB make new linear forms 2.
MKLF3 : UGB make new linear forms 3.
MKLIST : UGB make trace and cuts.
MKN0 : Make n0.

MKN1 : Make n1.
MKNEWP : Make new pairs.
MKNEWP : UGB make new critical pairs.
MKPAIR : Make pairs.
MKPAIR : UGB make critical pairs for polynomial list.
MKPOL : Make polynomial without green monomials.
MLDMKATOM : maslog demonstration make atomic formula.
MLDMKCNF : maslog demonstration make disjunctive normal form.
MLDMKDNF : maslog demonstration make disjunctive normal form.
MLDMKPOS : maslog demonstration make positive.
MLDMKPOS1 : maslog demonstration make positive 1.
MLDMKPRENEX : maslog demonstration make prenex.
MLDMKVD : maslog demonstration make variables disjoint.
pqmkaf : polynomial equation simplification make atomic formula.
PQMKCNF : polynomial equation make disjunctive normal form.
PQMKDNF : polynomial equation make disjunctive normal form.
PQMKPOS : polynomial equation make positive.
PQMKPRENEX : polynomial equation make prenex.
PQMKVD : polynomial equation make variable names disjoint.
tfmkaf : type formula make atomic formula.

mantissa

APMANT : Arbitrary precision floating point mantissa.

map

DEFMAP : Define generic map function.

mark

ADEPmark : Arbitrary domain extended polynomial mark.
DIDPALCMPC : Distributive domain polynomial array list check and mark polynomials.
DIIPALCMPC : Distributive integral polynomial array list check and mark polynomials.
PQBASE : symbol to mark a polynomial equation special atomic formula.

MAS

DOMAF : MAS Domain Algebraic Number Definition Module.
DOMAPF : MAS Domain Arbitrary Precision Floating Point Definition Module.
DOMC : MAS Domain Complex Number Definition Module.
DOMFF : MAS Domain Finite Field Definition Module.
DOMI : MAS Domain Integer Definition Module.
DOMIP : MAS Domain Integral Polynomial Definition Module.
DOMMD : MAS Domain Modular Digit Definition Module.
DOMMI : MAS Domain Modular Integer Definition Module.
DOMO : MAS Domain Octonion Number Definition Module.
DOMQ : MAS Domain Quaternion Number Definition Module.
DOMRF : MAS Domain Rational Function Definition Module.
DOMRN : MAS Domain Rational Number Definition Module.
DOMRP : MAS Domain Rational Polynomial Definition Module.
LINALGI : MAS Linear Algebra Integer Definition Module.

LINALGRN : MAS Linear Algebra Rational Number Definition Module.
MASADOM : MAS Arbitrary Domain Definition Module.
MASAPF : MAS Arbitrary Precision Floating Point Definition Module.
MASBIOS : MAS Basic I/O System Definition Module.
MASBIOSU : MAS BIOS Utility Definition Module.
MASC : MAS Complex Number Definition Module.
MASCHR : MAS character.
MASCOMB : MAS Combinatorial System Definition Module.
MASCONF : MAS Configuration Definition Module.
MASELEM : MAS Elementary Functions Definition Module.
MASERR : MAS Error Definition Module.
MASF : MAS Floating Point Definition Module.
MASFF : MAS Finite Field Definition Module.
MASI : MAS Integer Definition Module.
MASLISP : MAS Lisp Definition Module.
MASLISPU : MAS Lisp Utility Definition Module.
MASLOAD : MAS Load Definition Module.
MASLOADA : MAS Load Definition Module A.
MASLOADB : MAS Load Definition Module B.
MASLOADC : MAS Load Definition Module C.
MASLOADD : MAS Load Definition Module D.
MASLOADE : MAS Load Definition Module E.
MASLOADG : MAS Load Symmetric Functions Definition Module.
MASLOADJ : MAS Load Definition Module J.
MASLOADL : MAS Load Definition Module L.
MASLOADM : MAS Load Definition Module M.
MASLOADQ : MAS Load Definition Module Q.
MASLOADS : MAS Load Syzygy Definition Module.
MASmtc : MAS mtc [Modula-2 to C] Definition Module.
MASNC : MAS Non-commutative Product Definition Module.
MASNCC : MAS Non-commutative Center Definition Module.
MASNCGB : MAS Non-commutative Groebner Bases Definition Module.
MASO : MAS Octonion Number Definition Module.
MASORD : MAS order.
MASORDI : MAS order integer.
MASPARSE : MAS Parser Definition Module.
MASPGCD : MAS Polynomial GCD and RES System Definition Module.
MASQ : MAS Quaternion Number Definition Module.
masReadL : MAS Read Line.
masReadOpen : MAS Read Open
MASREP : MAS Representation Definition Module.
MASRN : MAS Rational Number Definition Module.
MASSET : MAS Set Definition Module.
massig : MAS Signal Handling Foreign Module.
MASSIGNAL : MAS Signal Handling Definition Module.
MASSPEC : MAS Specification Definition Module.
MASSTOR : MAS Storage Definition Module.
MASSYM : MAS Symbol Definition Module.
MASU : MAS Utility Definition Module.
MASYMDIP : MAS Symbol to DIP Definition Module.
MLMASLOG : MAS Logic Configuration Implementation Module.
MLMLDEMO : MAS Logic Demonstration Implementation Module.

mask

SigMask : Signal mask.
sigsetmask : Set signal mask.

masload

MLPQSMPL : Masload Polynomial Equation Simplify Definition Module.

maslog

InitExternalsMLDEMO : Initialize externals maslog demonstration procedures.
MASLOG : Maslog Definition Module.
MLDAPPLYAT : maslog demonstration apply to atomic formulas.
MLDCONTBDVAR : maslog demonstration contains bound variable.
MLDCONTVAR : maslog demonstration contain variable.
MLDEMO : Maslog Demonstration Definition Module.
MLDIREAD : maslog demonstration infix read.
MLDMKATOM : maslog demonstration make atomic formula.
MLDMKCNF : maslog demonstration make disjunctive normal form.
MLDMKDNF : maslog demonstration make disjunctive normal form.
MLDMKPOS : maslog demonstration make positive.
MLDMKPOS1 : maslog demonstration make positive 1.
MLDMKPRENEX : maslog demonstration make prenex.
MLDMKVD : maslog demonstration make variables disjoint.
MLDPPRT : maslog demonstration pretty print.
MLDPREAD : maslog demonstration prefix read.
MLDPREPQE : maslog demonstration prepare quantifier elimination.
MLDSMPL : maslog demonstration simplify.
MLDSUBSTVAR : maslog demonstration substitute variables.
MLDTEXW : maslog demonstration tex write.
MLDTST : maslog demonstration test 1.
MLOGBASE : Maslog Base Definition Module.
MLOGIO : Maslog Input Output System Definition Module.

MASSTOR

StorSummary : MASSTOR Summary.

matrices

RRADVARMATRICES : Real root arbitrary domain multiplication matrices of variables.
RRIVARMATRICES : Real root integral multiplication matrices of variables.

matrix

ADMPROD : Arbitrary domain matrix product.
ADMPTRACE : Arbitrary domain matrix product trace.
ADMSUM : Arbitrary domain matrix sum.
ADMTRACE : Arbitrary domain matrix trace.
ADMWRITE : Arbitrary domain matrix write.
ADSMPROD : Arbitrary domain scalar and matrix product.
ADUM : Arbitrary domain unit matrix.
DIPDEM : Distributive polynomial degree matrix.
DIPLDM : Distributive polynomial list degree matrix.

DMEVAD : Degree matrix exponent vector add.
EIMWRT : Exterior integral matrix write.
EXIDET : Exterior integral matrix determinant.
EXIDT2 : Exterior integral matrix determinant 2.
EXMHOM : Exterior matrix homomorphism.
ICHARPOL : Integral matrix characteristic polynomial.
IMDET : Integer matrix determinant, using Gaussian elimination.
IMDETL : Integer matrix determinant, using Laplace expansion.
IMDIF : Integer matrix difference.
IMFRNM : Integer matrix from rational number matrix.
IMFRNM : Integer matrix from rational number matrix.
IMFRNM1 : Integer matrix from rational number matrix.
IMFRNM1 : Integer matrix from rational number matrix.
IMGE : Integer matrix Gaussian elimination.
IMGELUD : Integer matrix Gaussian elimination LU-decomposition.
IMLT : Integer lower triangular matrix transformation.
IMMAX : Integer matrix maximum norm.
IMPROD : Integer matrix product.
IMPTRACE : Integral matrix product trace.
IMRTPROD : Integral matrix right tensor product.
IMSDS : Integer matrix solve decomposed system.
IMSUM : Integer matrix sum.
IMTRACE : Integral matrix trace.
IMUNS : Integer matrix upper triangular matrix solution null space.
IMUNS : Integer matrix upper triangular matrix solution null space.
IMUT : Integer upper triangular matrix transformation.
IMWRITE : Integer matrix write.
ISIG : Integral matrix signature.
ISMPROD : Integer scalar and matrix product.
IUM : Integer unit matrix.
MAIPDE : Matrix of integral polynomials determinant, exact division algorithm.
MAIPDM : Matrix of integral polynomials determinant, modular algorithm.
MAIPHM : Matrix of integral polynomials homomorphism.
MAIPP : Matrix of integral polynomials product.
MCPMV : Matrix of coefficients of polynomials, with respect to main variable.
MDELCOL : Matrix delete column.
MDIM : Matrix dimension.
MFILL : Matrix fill.
MGET : Matrix get.
MIAIM : Matrix of integers, adjoin identity matrix, A is an m by n matrix
MIAIM : Matrix of integers, adjoin identity matrix, A is an m by n matrix
MIAIM : Matrix of integers, adjoin identity matrix, A is an m by n matrix
MICINS : Matrix of integers column insertion.
MICS : Matrix of integers column sort.
MINNCT : Matrix of integers, non-negative column transformation.
MIRAND : Matrix random.
MMDDET : Matrix of modular digits determinant.
MMDNSB : Matrix of modular digits null space basis.
MMINOR : Matrix minor.
MMPDMA : Matrix of modular polynomials determinant, modular algorithm.
MMPEV : Matrix of modular polynomials evaluation.
MMULT : Matrix Multiplication.
MRANG : Matrix rang.

MREAD : Matrix Read.
MSET : Matrix set.
MTPLH : Matrix to Polynomial List Horizontal.
MTPLV : Matrix to Polynomial List Vertical.
MTRANS : Matrix transpose.
NLMMULT : Non-Commutative Matrix Multiplication.
NMREAD : Non-Commutative Matrix Read.
PLVTM : Polynomial List Vertical To Matrix.
PMWR : Polynomial Matrix Write.
PVDEMA : Permutation vector for degree matrix.
RNMDET : Rational number matrix determinant, using Gaussian elimination.
RNMDETL : Rational number matrix determinant, using Laplace expansion.
RNMDIF : Rational number matrix difference.
RNMFIM : Rational number matrix from integer matrix.
RNMFIM : Rational number matrix from integer matrix.
RNMGE : Rational number matrix Gaussian elimination.
RNMGELUD : Rational number matrix Gaussian elimination LU-decomposition.
RNMHILBERT : Rational number matrix Hilbert.
RNMINV : Rational number matrix inversion.
RNMINVI : Rational number matrix inversion, integer algorithm.
RNMLT : Rational matrix lower triangular matrix transformation.
RNMLT : Rational matrix lower triangular matrix transformation.
RNMMAX : Rational number matrix maximum norm.
RNMPROD : Rational number matrix product.
RNMREAD : Rational number matrix read.
RNMSDS : Rational number matrix solve decomposed system.
RNMSUM : Rational number matrix sum.
RNMUNS : Rational number matrix upper triangular matrix solution null space.
RNMUNS : Rational number matrix upper triangular matrix solution null space.
RNMUT : Rational matrix upper triangular matrix transformation.
RNMUT : Rational matrix upper triangular matrix transformation.
RNMWRITE : Rational number matrix write.
RNSMPROD : Rational number scalar and matrix product.
RNUM : Rational number unit matrix.
RRADPOLMATRIX : Real root arbitrary domain polynomial matrix.
RRIPOLMATRIX : Real root integral polynomial matrix.
RRMMULT : Real root matrix of multiplication.
VMADD : Vertical Matrix Addition.

maximal

DIDIMS : Distributive polynomial dimension maximal independent set.
DIMIS : Dimension and maximal independent set.

maximum

DIIPMN : Distributive integral polynomial maximum norm.
DIPLMD : distributive polynomial list maximum degree.
DIRPMN : Distributive rational polynomial maximum norm.
IMAX : Integer maximum.
IMMAX : Integer matrix maximum norm.
IPMAXN : Integral polynomial maximum norm.
IVMAX : Integer vector maximum norm.
MASMAX : Maximum.
RNMAX : Rational number maximum.
RNMMAX : Rational number matrix maximum norm.
RNVMAX : Rational number vector maximum norm.
VMAX : Vector maximum.
VMIN : Vector maximum.

medium

IMPDS : Integer medium prime divisor search.

member

COMPA1 : UGB trace member in trace list.
WMEMB : White term member.

membership

IdealMember : ideal membership test.
MEMBER : Membership test.
MEMQ : Membership test equal pointers.
NFEXEC : Parametric ideal membership exec.
RadicalMember : radical membership test.
SMEMB : Symbol membership.
SMEMB : Symbol membership.
WRQFN0 : Write quantifier free formula for parametric ideal membership.

merge

CGBLM : CGB coloured distributive polynomial list merge.
CGBLPM : CGB list merge.
DIDPCPLMS1 : Distributive domain polynomial list construct pairs list merge sort.
DIDPLM1 : Distributive domain polynomial list merge sort.
DIIPCPLMS1 : Distributive integral polynomial list construct pairs list merge sort.
DIIPLM1 : Distributive integral polynomial list merge sort.
DIPLM : Distributive polynomial list merge.
EVPLM : Exponent vector pair-list merge.
EVUMSORT : Exponent vector unique merge sort.
IPFLMER : integral polynomial factorlist merge.
LBIM : List of beta-integers merge.
LMERGE : List merge.
LRNM : List of rational numbers merge.
MERGE : Noether Merge of Base Polynomials.
MERGE : UGB merge stacks.

method

IPRCNP : Integral polynomial real root calculation, newton method preparation.
IPRIM : Integral polynomial real root isolation, modified Uspensky method.
IPRIMO : Integral polynomial real root isolation, modified Uspensky method, open interval.
IPRIMS : Integral polynomial real root isolation, modified Uspensky method, standard interval.
IPRIMU : Integral polynomial real root isolation, modified Uspensky method, unit interval.
IPRIU : Integral polynomial real root isolation, Uspensky method.
IPRIUP : Integral polynomial real root isolation, Uspensky method, positive roots.
IPSRM : Integral polynomial strong real root isolation, modified Uspensky method.
IPSRMS : Integral polynomial strong real root isolation, modified Uspensky method, standard interval.

milliseconds

ClocK : ClocK returns milliseconds of the processes cpu-time.

minimal

AFPMPR : Algebraic number field polynomial minimal polynomial of a real root.
DIGBMI : Distributive minimal ordered groebner basis.
DIGMIN : Distributive minimal ordered groebner basis.
DIIFMI : Distributive minimal ordered groebner basis.
DIIGMI : Distributive minimal ordered groebner basis.
DINLGM : Distributive non-commutative minimal ordered left Groebner base.
DINLMPG : Distributive non-commutative left rational minimal polynomial for a G basis.
DINLMPL : Distributive non-commutative left rational minimal polynomial list for a G basis.
DIPMVV : distributive polynomial minimal variable vector.
DIPSSM : Distributive polynomial sort and select minimal.
DIRMPG : Distributive rational minimal polynomial for a groebner basis.
IUPMRN : Integral univariate polynomial minimal polynomial of a rational number.
RUPMRN : Rational univariate polynomial minimal polynomial of a rational number.

minimize

MINPP : Minimize polynomials list.

minimum

IMIN : Integer minimum.
MASMIN : Minimum.

minor

MMINOR : Matrix minor.

minus

SetMinus : Set minus.
SetMinusC : Set minus constructive.
SetMinusCQ : Set minus constructive equal.
SetMinusQ : Set minus equal.

miscellaneous

CGBMISC : Comprehensive-Groebner-Bases Miscellaneous Programs Definition Module.
SetUnion : Miscellaneous CGB Functions.

mod

IPIHOM : Integral polynomial mod ideal homomorphism.
IPIPR : Integral polynomial mod ideal product.
IPIQH : Integral polynomial mod ideal quadratic Hensel lemma.
MIPIPR : Modular integral polynomial mod ideal product.
MIPISE : Modular integral polynomial mod ideal, solution of equation.
MMPIQR : Modular monic polynomial mod ideal quotient and remainder.
MPIQH : Modular polynomial mod ideal, quadratic Hensel lemma.
MPIQHL : Modular polynomial mod ideal quadratic Hensel lemma, list.
MPIQHS : Modular polynomial mod ideal, quadratic Hensel lemma on a single variable.

model

DMIA : Define model, implementation or axioms.

modified

IPRIM : Integral polynomial real root isolation, modified Uspensky method.
IPRIMO : Integral polynomial real root isolation, modified Uspensky method, open interval.
IPRIMS : Integral polynomial real root isolation, modified Uspensky method, standard interval.
IPRIMU : Integral polynomial real root isolation, modified Uspensky method, unit interval.
IPSRM : Integral polynomial strong real root isolation, modified Uspensky method.
IPSRMS : Integral polynomial strong real root isolation, modified Uspensky method, standard interval.
LDSMKB : Linear diophantine system solution, modified Kannan and Bachem algorithm.
PMDEG : Polynomial modified degree.

modul

MGB : Modul Groebner Base.
NLMGB : Non-Commutative Modul Groebner Base.

modula

InitExternalsG : Tell Modula and LISP about external compiled procedures.
InitExternalsS : Tell Modula and LISP about external compiled procedures.
MODVAR : Modula Global Variable Implementation Module.
MVDeclareB : modula variable declare boolean.
MVDeclareL : modula variable declare list.
MVFLAG : modula variable get.
MVGET : modula variable get.
MVHLP : modula variable help.
MVOFF : modula variable off.
MVON : modula variable on.
MVSET : modula variable set.
MWRIT1 : Output in modula like syntax.
MWRITE : Output in modula like syntax.

modular

DMPPRD : Dense modular polynomial product.
DMPSUM : Dense modular polynomial sum.
DMUPNR : Dense modular univariate polynomial natural remainder.
DomLoadMD : Domain load modular digit.
DomLoadMI : Domain load modular integer.
DOMMD : MAS Domain Modular Digit Definition Module.
DOMMI : MAS Domain Modular Integer Definition Module.
MAIPDM : Matrix of integral polynomials determinant, modular algorithm.
MDCRA : Modular digit chinese remainder algorithm.
MDDIF : Modular digit difference.
MDEXP : Modular digit exponentiation.
MDHOM : Modular digit homomorphism.
MDINV : Modular digit inverse.
MDLCRA : Modular digit list chinese remainder algorithm.
MDNEG : Modular digit negative.
MDPROD : Modular digit product.
MDQ : Modular digit quotient.
MDRAN : Modular digit, random.
MDSUM : Modular digit sum.
MDVHOM : Modular vector homomorphism.
MIDCRA : Modular integer digit chinese remainder algorithm.
MIDIF : Modular integer difference.
MIEXP : Modular integer exponentiation.
MIHOM : Modular integer homomorphism.
MIINV : Modular integer inverse.
MINEG : Modular integer negation.
MIPDIF : Modular integral polynomial difference.
MIPFSM : Modular integral polynomial from symmetric modular.
MIPFSM : Modular integral polynomial from symmetric modular.
MIPHOM : Modular integral polynomial homomorphism.
MIPIPR : Modular integral polynomial mod ideal product.
MIPISE : Modular integral polynomial mod ideal, solution of equation.
MIPNEG : Modular integral polynomial negation.
MIPPR : Modular integral polynomial product.
MIPRAN : Modular integral polynomial, random.
MIPROD : Modular integer product.

MIPSUM : Modular integral polynomial sum.
MIQ : Modular integer quotient.
MIRAN : Modular integer, random.
MISUM : Modular integer sum.
MIUPQR : Modular integral univariate polynomial quotient and remainder.
MIUPSE : Modular integral univariate polynomial, solution of equation.
MMDDET : Matrix of modular digits determinant.
MMDNSB : Matrix of modular digits null space basis.
MMPDMA : Matrix of modular polynomials determinant, modular algorithm.
MMPDMA : Matrix of modular polynomials determinant, modular algorithm.
MMPEV : Matrix of modular polynomials evaluation.
MMPIQR : Modular monic polynomial mod ideal quotient and remainder.
MPDIF : Modular polynomial difference.
MPEMV : Modular polynomial evaluation of main variable.
MPEVAL : Modular polynomial evaluation.
MPEXP : Modular polynomial exponentiation.
MPGCDC : Modular polynomial greatest common divisor and cofactors.
MPHOM : Modular polynomial homomorphism.
MPINT : Modular polynomial interpolation.
MPIQH : Modular polynomial mod ideal, quadratic Hensel lemma.
MPIQHL : Modular polynomial mod ideal quadratic Hensel lemma, list.
MPIQHS : Modular polynomial mod ideal, quadratic Hensel lemma on a single variable.
MPMDP : Modular polynomial modular digit product.
MPMDP : Modular polynomial modular digit product.
MPMON : Modular polynomial monic.
MPNEG : Modular polynomial negative.
MPPROD : Modular polynomial product.
MPPSR : Modular polynomial pseudo-remainder.
MPQ : Modular polynomial quotient.
MPQR : Modular polynomial quotient and remainder.
MPRAN : Modular polynomial, random.
MPRES : Modular polynomial resultant.
MPSPRS : Modular polynomial subresultant polynomial remainder sequence.
MPSUM : Modular polynomial sum.
MPUC : Modular polynomial univariate content.
MPUCPP : Modular polynomial univariate content and primitive part.
MPUCS : Modular polynomial univariate content subroutine.
MPUP : Modular polynomial univariate product.
MPUPP : Modular polynomial univariate primitive part.
MPUQ : Modular polynomial univariate quotient.
MUPBQP : Modular univariate polynomial Berlekamp q polynomials construction.
MUPDDF : Modular univariate polynomial distinct degree factorization.
MUPDER : Modular univariate polynomial derivative.
MUPEGC : Modular univariate polynomial extended greatest common divisor.
MUPFBL : Modular univariate polynomial factorization-Berlekamp algorithm.
MUPFS : Modular univariate polynomial factorization, special.
MUPGCD : Modular univariate polynomial greatest common divisor.
MUPHEG : Modular univariate polynomial half-extended greatest common divisor.
MUPRAN : Modular univariate polynomial, random.
MUPRC : Modular univariate polynomial resultant and cofactor.
MUPRES : Modular univariate polynomial resultant.

MUPSFF : Modular univariate polynomial squarefree factorization.
SACM : SAC Modular Digit and Integer Definition Module.
SACMPOL : SAC Modular Polynomial Definition Module.
SACMUFAC : SAC Modular Univariate Polynomial Factorization Definition Module.
SMFMI : Symmetric modular from modular integer.
SMFMI : Symmetric modular from modular integer.
SMFMIP : Symmetric modular from modular integral polynomial.
SMFMIP : Symmetric modular from modular integral polynomial.
VMPIP : Vector of modular polynomial inner product.

module

ADEXTRA : Arbitrary domain extra definition module.
ADTOOLS : Arbitrary Domain Tools Definition Module.
ALDPARSE : Aldes Parser Definition Module.
CGBAPPL : Comprehensive-Groebner-Bases Applications Definition Module.
CGBDSTR : Comprehensive-Groebner-Bases Data-Structures Definition Module.
CGBFUNC : Comprehensive-Groebner-Bases Utility Functions Definition Module.
CGBMAIN : Comprehensive-Groebner-Bases Main Programms Definition Module.
CGBMISC : Comprehensive-Groebner-Bases Miscellaneous Programs Definition Module.
CGBSYS : Comprehensive-Groebner-Bases System Definition Module.
clock : clock Foreign Module.
DIPADOM : DIP Arbitrary Domain Definition Module.
DIPAGB : DIP Arbitrary Domain Groebner Basis Definition Module.
DIPC : DIP Common Polynomial System Definition Module.
DIPCJ : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPDCGB : DIP Decompositional Groebner Bases Definition Module.
DIPDCIB : DIP Decompositional Involutive Bases Definition Module.
DIPDDGB : DIP Domain D-Groebner Bases Definition Module.
DIPDEC0 : DIP Ideal Decomposition 0 System Definition Module.
DIPDIM : DIP Dimension Definition Module.
DIPE : DIP Exterior Algebra Definition Module.
DIPGB : DIP Groebner Bases Definition Module.
DIPGCD : DIP GCD Definition Module.
DIPI : DIP Integral Definition Module.
DIPIB : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPIDEAL : DIP Ideal System Definition Module.
DIPIDGB : DIP Integral D-Groebner Bases Definition Module.
DIPIGB : DIP Integral Groebner Bases Definition Module.
DIPIIB : DIP Integral Polynomial System Definition Module in the sense of Janet.
DIPIPOL : DIP Integer Polynomial Definition Module.
DIPRF : DIP Rational Function Definition Module.
DIPRN : DIP Rational Definition Module.
DIPRNGB : DIP Rational Groebner Bases Definition Module.
DIPRNIB : DIP Rational Numbers Polynomial Definition Module in the sense of Janet.
DIPRNPOL : DIP Rational Number Polynomial Definition Module.
DIPROOT : DIP Ideal Real Root System Definition Module.
DIPSP : DIP Integral Function Groebner Bases Implementation Module.
DIPTOO : DIP Termorder Optimization Definition Module.

DIPTOOLS : Distributive Polynomials Tools Definition Module.
DIPZ : DIP Zero Dimensional Ideal Definition Module.
DOMAF : MAS Domain Algebraic Number Definition Module.
DOMAPF : MAS Domain Arbitrary Precision Floating Point Definition Module.
DOMC : MAS Domain Complex Number Definition Module.
DOMFF : MAS Domain Finite Field Definition Module.
DOMI : MAS Domain Integer Definition Module.
DOMIP : MAS Domain Integral Polynomial Definition Module.
DOMMD : MAS Domain Modular Digit Definition Module.
DOMMI : MAS Domain Modular Integer Definition Module.
DOMO : MAS Domain Octonion Number Definition Module.
DOMQ : MAS Domain Quaternion Number Definition Module.
DOMRF : MAS Domain Rational Function Definition Module.
DOMRN : MAS Domain Rational Number Definition Module.
DOMRP : MAS Domain Rational Polynomial Definition Module.
GSYMFUIN : G-Symmetric Integral Polynomial System Definition Module.
GSYMFURN : G-Symmetric Rational Polynomial System Definition Module.
GSYMINP : GSYM Input Definition Module.
LINALG : Linear algebra definition module.
LINALGI : MAS Linear Algebra Integer Definition Module.
LINALGRN : MAS Linear Algebra Rational Number Definition Module.
LISTTOOLS : List Tools Definition Module.
MASADOM : MAS Arbitrary Domain Definition Module.
MASAPF : MAS Arbitrary Precision Floating Point Definition Module.
MASBIOS : MAS Basic I/O System Definition Module.
MASBIOSU : MAS BIOS Utility Definition Module.
MASC : MAS Complex Number Definition Module.
MASCOMB : MAS Combinatorial System Definition Module.
MASCONF : MAS Configuration Definition Module.
MASELEM : MAS Elementary Functions Definition Module.
MASERR : MAS Error Definition Module.
MASF : MAS Floating Point Definition Module.
MASFF : MAS Finite Field Definition Module.
MASI : MAS Integer Definition Module.
maskpathsea : kpathsearch Foreign Module.
MASLISP : MAS Lisp Definition Module.
MASLISPU : MAS Lisp Utility Definition Module.
MASLOAD : MAS Load Definition Module.
MASLOADA : MAS Load Definition Module A.
MASLOADB : MAS Load Definition Module B.
MASLOADC : MAS Load Definition Module C.
MASLOADD : MAS Load Definition Module D.
MASLOADE : MAS Load Definition Module E.
MASLOADG : MAS Load Symmetric Functions Definition Module.
MASLOADJ : MAS Load Definition Module J.
MASLOADL : MAS Load Definition Module L.
MASLOADM : MAS Load Definition Module M.
MASLOADQ : MAS Load Definition Module Q.
MASLOADS : MAS Load Syzygy Definition Module.
MASLOG : Maslog Definition Module.
MASmtc : MAS mtc [Modula-2 to C] Definition Module.
MASNC : MAS Non-commutative Product Definition Module.
MASNCC : MAS Non-commutative Center Definition Module.

MASNCGB : MAS Non-commutative Groebner Bases Definition Module.
MASO : MAS Octonion Number Definition Module.
MASPARSE : MAS Parser Definition Module.
MASPGCD : MAS Polynomial GCD and RES System Definition Module.
MASQ : MAS Quaternion Number Definition Module.
masreadline : readline Foreign Module.
MASREP : MAS Representation Definition Module.
MASRN : MAS Rational Number Definition Module.
MASSET : MAS Set Definition Module.
massig : MAS Signal Handling Foreign Module.
MASSIGNAL : MAS Signal Handling Definition Module.
MASSPEC : MAS Specification Definition Module.
MASSTOR : MAS Storage Definition Module.
MASSYM : MAS Symbol Definition Module.
MASSYM2 : MAS/SAC Symbol System Definition Module 2.
MASU : MAS Utility Definition Module.
MASUGB : Universal Groebner Bases Definition Module.
MASYMDIP : MAS Symbol to DIP Definition Module.
MLDEMO : Maslog Demonstration Definition Module.
MLMASLOG : MAS Logic Configuration Implementation Module.
MLMLDEMO : MAS Logic Demonstration Implementation Module.
MLOGBASE : Maslog Base Definition Module.
MLOGIO : Maslog Input Output System Definition Module.
MLPQSMPL : Masload Polynomial Equation Simplify Definition Module.
MODVAR : Modula Global Variable Implementation Module.
NOETHER : Noether Polynomial System Definition Module.
Portab : Portability Definition Module.
PQBASE : Polynomial Equation Base Definition Module.
PQSMPL : Polynomial Equation Simplification Definition Module.
RRADOM : Real Root Arbitrary Domain Definition Module.
RRINT : Real Root Integral Definition Module.
RRUADOM : Real Root Univariate Arbitrary Domain Definition Module.
RRUINT : Real Root Univariate Integral Definition Module.
SACANF : SAC Algebraic Number Field Definition Module.
SACBIOS : SAC Basic I/O System Definition Module.
SACCOMB : SAC Combinatorical System Definition Module.
SACD : SAC Digit Definition Module.
SACDPOL : SAC Dense Polynomial Definition Module.
SACEXT1 : SAC Extensions 1 Definition Module.
SACEXT2 : SAC Extensions 2 Definition Module.
SACEXT3 : SAC Extensions 3 Definition Module.
SACEXT4 : SAC Extensions 4 Definition Module.
SACEXT5 : SAC Extensions 5 Definition Module.
SACEXT6 : SAC Extensions 6 Definition Module.
SACEXT7 : SAC Extensions 7 Definition Module.
SACEXT8 : SAC Extensions 8 Definition Module.
SACI : SAC Integer Definition Module.
SACIPOL : SAC Integer Polynomial System Definition Module.
SACLDIO : SAC Linear Diophantine Equation System Definition Module.
SACLIST : SAC List Processing Definition Module.
SACM : SAC Modular Digit and Integer Definition Module.
SACMPOL : SAC Modular Polynomial Definition Module.
SACMUFAC : SAC Modular Univariate Polynomial Factorization Definition Module.

SACPFAC : SAC Polynomial Factorization Definition Module.
SACPGCD : SAC Polynomial GCD and RES System Definition Module.
SACPOL : SAC Polynomial System Definition Module.
SACPRIM : SAC Factorization and Prime Number Definition Module.
SACRN : SAC Rational Number Definition Module.
SACROOT : SAC Polynomial Real Root Definition Module.
SACRPOL : SAC Rational Polynomial Definition Module.
SACSET : SAC Set Definition Module.
SACSYM : SAC Symbol System Definition Module.
SACSYM2 : SAC Symbol 2 Definition Module.
SACUPFAC : SAC Univariate Polynomial Factorization Definition Module.
setjmp : Setjmp Foreign Module.
SUBST : Substitution Group Polynomial System Definition Module.
SYMMFU : Symmetric Functions Definition Module.
SYSINFO : System Informations Definition Module.
SYZFUNC : Syzygy Functions Definition Module.
SYZGB : Syzygy Groebner Base Definition Module.
SYZHLP : Syzygy Utility Programs Definition Module.
SYZMAIN : Syzygy Main Programs Definition Module.
TFORM : Type Formula Definition Module.
TIPRNGB : DIP Rational Extended Groebner Bases Definition Module.

modulo

ADGJredI : Arbitrary domain polynomial G Janet-reducible modulo I.
ADIredG : Arbitrary domain polynomial set I reducible modulo G.
ADPNFJS : Arbitrary domain polynomial normalform in the sense of Janet modulo a set
DIPINFJS : Integral Distributive polynomial normal form in the sense of Janet modulo a
DIPNFJS : Distributive polynomial normal form in the sense of Janet modulo a set of

modulus

FRLSM : Fermat residue list, single modulus.

monic

AFPMON : Algebraic number field polynomial monic.
DILMOC : distributive polynomial monic.
DIPMOC : Distributive polynomial monic.
DIRPMC : Distributive rational polynomial monic.
MMPIQR : Modular monic polynomial mod ideal quotient and remainder.
MPMON : Modular polynomial monic.
RPMAIP : Rational polynomial monic associate of integral polynomial.
RPMON : Rational polynomial monic.

monomial

DILIMO : distributive polynomial list inverse monomial order.
DIMPAD : distributive monomial partial derivation.
DIPEVL : Distributive polynomial exponent vector leading monomial.
DIPFMO : Distributive polynomial from monomial.
DIPIMO : distributive polynomial inverse monomial order.
DIPMAD : Distributive polynomial monomial advance.
DIPMCP : Distributive polynomial monomial composition.
DIPMRD : Distributive polynomial monomial reductum.
DIPMST : Distributive polynomial monomial set.
EDIPEVL : Extended distributive polynomial exponent vector of the leading monomial.
FINDRM : Find monomial.
MPPFMP : monomial of polynomial ring over polynomial ring from monomial of
MPPFMP : monomial of polynomial ring over polynomial ring from monomial of
PMON : Polynomial monomial.

monomials

CGBCOL : Write coloured polynomials without green monomials.
DIPXCM : distributive polynomial extract constant monomials.
GGREEN : Write groebner-system without green monomials.
GREPOL : Get polynomials without green monomials.
MKPOL : Make polynomial without green monomials.

mtc

MASmtc : MAS mtc [Modula-2 to C] Definition Module.

multilinear

GSYMLT : G-Symmetric Multilinear Terms.

multiple

ADLCM : Arbitrary domain least common multiple.
AFPME : Algebraic number field, polynomial multiple evaluation.
DIGISM : DIP G base index search for extension multiple univariats.
ECPLCMHT : Extended critical pair select the least common multiple of head terms.
EVLCM : Exponent vector least common multiple.
EVMT : Exponent vector multiple test.
EVMTJ : Exponent vector multiple test in the sense of Janet.
ILCM : Integer least common multiple.
IPAFME : Integral polynomial, algebraic number field multiple evaluation.
IPLCM : Integral polynomial least common multiple.
RPAFME : Rational polynomial, algebraic number field multiple evaluation.
RPBLGS : Rational polynomial base coefficients least common multiple, greatest common divisor, and sign.
RUPLCM : Rational univariate polynomial least common multiple.

multiplication

CMULT	: Colour multiplication.
DIPMPM	: Distributive polynomial multiplication by power of main variable.
DIPMPV	: Distributive polynomial multiplication by power of variable.
DIRPMV	: Distributiv Polynomial multiplication with a variable.
IMP2	: Integer multiplication by power of 2.
MMULT	: Matrix Multiplication.
NLMMULT	: Non-Commutative Matrix Multiplication.
NLPLMULT	: Non-Commutative Polynomial List Multiplication.
NOEPIP	: Noether Polynomial Factor Multiplication.
PLMULT	: Polynomial List Multiplication.
PMPMV	: Polynomial multiplication by power of main variable.
PMPV	: Polynomial multiplication by power of variable.
RRADVARMATRICES	: Real root arbitrary domain multiplication matrices of variables.
RRIVARMATRICES	: Real root integral multiplication matrices of variables.
RRMMULT	: Real root matrix of multiplication.

n-th

APROOT	: Arbitrary precision floating point n-th root.

n0

GSYSN0	: Groebner-System n0 update.
MKN0	: Make n0.

n1

MKN1	: Make n1.

name

ARRAYDEC	: Generate array name declarations.
EDIT	: Edit file with name s.
NAME	: Name.
NAME	: Name.

names

FORMKVD	: formula make variable names disjoint.
PQMKVD	: polynomial equation make variable names disjoint.

natural

DMUPNR	: Dense modular univariate polynomial natural remainder.

negate

pqnegaf	: negate atomic formula.

negation

AFNEG : Algebraic number field element negation.
FFNEG : Finite field negation.
INEG : Integer negation.
MINEG : Modular integer negation.
MIPNEG : Modular integral polynomial negation.
SetNegFunc : Set negation function in domain.
VINEG : Vector of integers negation.

negative

ADNEG : Arbitrary domain negative.
ADPNEG : Arbitrary domain polynomial negative.
AFPNEG : Algebraic number field polynomial negative.
APNEG : Arbitrary precision floating point negative.
CNEG : Complex number negative.
DIIPNG : Distributive integral polynomial negative.
DIPNEG : Distributive polynomial negative.
DIRPNG : Distributive rational polynomial negative.
EIVNEG : Exterior integral vector negative.
IPNEG : Integral polynomial negative.
IUPNT : Integral univariate polynomial negative transformation.
MDNEG : Modular digit negative.
MPNEG : Modular polynomial negative.
ONEG : Octonion number negative.
QNEG : Quaternion number negative.
RFNEG : Rational function negative.
RNNEG : Rational number negative.
RPNEG : Rational polynomial negative.

new

DILINV : distributive polynomial list introduce new variable.
DIPINV : Distributive polynomial introduction of new variables.
EVINV : Exponent vector introduction of new variables.
EVLINV : Exponent vector list introduction of new variables.
ISNEU : UGB new terms test.
ISNEUL : UGB new linear form test.
MKCOL : Make new colour.
MKLF1 : UGB make new linear forms 1.
MKLF2 : UGB make new linear forms 2.
MKLF3 : UGB make new linear forms 3.
MKNEWP : Make new pairs.
MKNEWP : UGB make new critical pairs.
NEULF : UGB compute new linear forms from new terms.
NEULF : UGB compute new linear forms from new terms.
NewDom : New domain.
NEWL : UGB update linear forms from new terms.
NEWQUE : New Queue.
NewRep : New representation.
NONEWL : UGB update linear forms without new terms.
PINV : Polynomial introduction of new variables.
SYONP : Syzygy for old Polynomials by new Polynomials.

newton

IPRCNP : Integral polynomial real root calculation, newton method preparation.

next

CPLEXN : Cartesian product, lexicographically next.
LEXNEX : Lexicographically next.
MKSP1 : UGB compute next non-zero reduced S-polynomial.
NEXTPAIR : Next Pair of Polynomials.
NextParm : Next Parameter.
NextRel : next relation.
PARTN : Partition, next.
TfNextTuple : type formula next tuple.

nodes

STNLST : Symbol tree nodes list.

noether

MERGE : Noether Merge of Base Polynomials.
NOEINF : Noether Polynomial System Information.
NOEL32 : Noether SK Polynomial Computation.
NOENSP : Noether Number of Noetherian Base Polynomials.
NOEPIP : Noether Polynomial Factor Multiplication.
NOEPOW : Noether SK Power Sum Computation.
NOEPPR : Noether Polynomial Product.
NOEPRM : Noether Remove.
NOEPSM : Noether Polynomial Sum.
NOERED : Noether G-Symmetric Polynomial Computation.
NOESRT : Noether Polynomial Sort.
NOETHER : Noether Polynomial System Definition Module.
SUBPOW : Noether SK Power Sum Computation for Substitution Groups.
SUBRED : Noether G-Symmetric Polynomial Computation for Substitution Groups.

noetherian

DINNGB : DIP Groebner bases for non noetherian polynomial rings.
NOENSP : Noether Number of Noetherian Base Polynomials.

nomalized

DINTZS : DIP nomalized tupels from system zero.

non

DINNGB : DIP Groebner bases for non noetherian polynomial rings.
EVNNZE : Exponent vector number of non zero exponents.

non-commutative

DIN1GB : Distributive non-commutative polynomials Groebner base.
DINCCO : Distributive rational non-commutative polynomial commutator.
DINCCP : Distributive rational non-commutative polynomial center polynomial.
DINCCPpre : Distributive rational non-commutative polynomial center polynomial preparation.
DINCGB : Distributive non-commutative polynomials Groebner base.
DINLGB : Distributive non-commutative polynomials left Groebner base.
DINLGM : Distributive non-commutative minimal ordered left Groebner base.
DINLIS : Distributive non-commutative polynomial list left irreducible set.
DINLMPG : Distributive non-commutative left rational minimal polynomial for a G basis.
DINLMPL : Distributive non-commutative left rational minimal polynomial list for a G basis.
DINLNF : Distributive non-commutative polynomial left normal form.
DINLRD : Distributive non-commutative polynomial list read.
DINLSP : Distributive non-commutative polynomial left S-polynomial.
DINPEX : Distributive non-commutative polynomial exponentiation.
DINPPR : Distributive polynomial non-commutative product.
DINPQ : Distributive non-commutative Polynomial Quotient.
DINPRD : Distributive non-commutative polynomial read.
DINPTL : Distributive polynomial non-commutative product table lookup.
DINPTslT : Distributive polynomial non-commutative product table strict lex test.
DINPTU : Distributive polynomial non-commutative product table update.
InitExternalsC : Initialize external compiled non-commutative polynomial procedures.
MASNC : MAS Non-commutative Product Definition Module.
MASNCC : MAS Non-commutative Center Definition Module.
MASNCGB : MAS Non-commutative Groebner Bases Definition Module.

non-Commutative

NLBGFUP : Non-Commutative Base Generators Factor Update.
NLDGBRED : Non-Commutative Discrete Groebner Base Reduction.
NLGBE : Non-Commutative Groebner Base with Exponent Vector Check.
NLGBEF : Non-Commutative Groebner Base with Exponent Vector Check and Factors.
NLGBF : Non-Commutative Groebner Base with Factors.
NLHEQ : Non-Commutative Homogenous Equation.
NLHSEQ : Non-Commutative Homogenous System of Equation.

non-commutative

NLIEQ : Special Solution for inhomogenous non-commutative equation.
NLISEQ : Special Solution for inhomogenous non-commutative system of equation.

non-Commutative

NLMGB : Non-Commutative Modul Groebner Base.
NLMMULT : Non-Commutative Matrix Multiplication.
NLPLMULT : Non-Commutative Polynomial List Multiplication.
NLRCSP : Non-Commutative Reduction Chain of S-Polynomials.
NLSPC : Non-Commutative S-Polynomial with Coefficients.
NLSPCEGB : Non-Commutative S-Polynomials with Coefficients and Exponenetvector-Check
NLSPCGB : Non-Commutative S-Polynomials with Coefficients for Groebner Base.
NMREAD : Non-Commutative Matrix Read.

non-commutative

SINL : Special Solution for inhomogenous non-commutative system of equation.
STINL : Solution Test for inhomogenous non-commutative Case.
SYHNL : Syzygy for homogenous non-commutative system of equation.
SYTHNL : Syzygy Test for homogenous non-commutative Case.

non-generic

SwitchParse : Switch parsing between generic / non-generic parse.

non-implication

DNIMP : Digit non-implication.

non-negative

DRANN : Digit, random non-negative.
MINNCT : Matrix of integers, non-negative column transformation.

non-noetherian

DNLCPL : distributive polynomial non-noetherian left construct pair list.
DNLUPL : distributive polynomial non-noetherian left update pair list.
DNN2GB : distributive polynomials non-noetherian 2-sided Groebner base.
DNNLGB : distributive non-noetherian polynomials left Groebner base.
DNNRGB : distributive polynomials non-noetherian right Groebner base.
DNRCPL : distributive polynomial non-noetherian right construct pair list.
DNRUPL : distributive polynomial non-noetherian right update pair list.

non-zero

CondNzero : Condition non-zero part.
MKSP1 : UGB compute next non-zero reduced S-polynomial.

nonmultiple

DIPNML : Distributive polynomial nonmultiple variable list.

norm

DIIPMN : Distributive integral polynomial maximum norm.
DIIPNORM : Distributive integral polynomial norm.
DIIPSN : Distributive integral polynomial sum norm.
DIITNT : Distributive polynomial system interval tupel from norm tupel.
DIRPMN : Distributive rational polynomial maximum norm.
DIRPSN : Distributive rational polynomial sum norm.
IMMAX : Integer matrix maximum norm.
IPMAXN : Integral polynomial maximum norm.
IPSUMN : Integral polynomial sum norm.
IVMAX : Integer vector maximum norm.
RNMMAX : Rational number matrix maximum norm.
RNVMAX : Rational number vector maximum norm.

normal

ADPSUGNF : Arbitrary domain polynomial normal with sugar strategy normalform.
ADPSUGSP : Arbitrary domain polynomial normal with sugar strategy S-polynomial.
DIIFNF : Distributive integral function polynomial normal form.
DIIPDNF : Distributive integral polynomial normal form.
DIIPNF : Distributive integral polynomial normal form.
DIIPNFJ : Distributive integral polynomial normal form in the sense of Janet.
DILADNF : distributive polynomial list arbitrary domain normal form.
DINLNF : Distributive non-commutative polynomial left normal form.
DIPADNF : distributive polynomial arbitrary domain normal form.
DIPEINFJ : Integral distributive extended polynomial normal form in the sense of Janet.
DIPENFJ : Distributive extended polynomial normal form in the sense of Janet.
DIPINFJ : Integral Distributive polynomial normal form in the sense of Janet.
DIPINFJS : Integral Distributive polynomial normal form in the sense of Janet modulo a
DIPNFJ : Distributive polynomial normal form in the sense of Janet.
DIPNFJS : Distributive polynomial normal form in the sense of Janet modulo a set of
DIPNOR : Distributive polynomial normal form.
DIRPNF : Distributive rational polynomial normal form.
DIRPNFJ : Distributive rational polynomial normal form in the sense of Janet.
EDIIFSUGNF : Extended distributive integral function polynomial normal with sugar
EDIIFSUGSP : Extended distributive integral function polynomial normal with sugar
EDIPNOR : Extended distributive polynomial normal form.
EDIPSUGCON : Extended distributive polynomial normal with sugar strategy constructor.
EDIPSUGNOR : Extended distributive polynomial normal with sugar strategy normal form.
EDIPSUGNOR : Extended distributive polynomial normal with sugar strategy normal form.
EDIPSUGSP : Extended distributive polynomial normal with sugar strategy S-polynomial.
MLDMKCNF : maslog demonstration make disjunctive normal form.
MLDMKDNF : maslog demonstration make disjunctive normal form.
PFIDNOR : Integral Polynomial D Normal Form.
PFILNOR : Integral Polynomial List Normal Form.
PFINOR : Integral Polynomial Normal Form.
PQMKCNF : polynomial equation make disjunctive normal form.
PQMKDNF : polynomial equation make disjunctive normal form.
PQSCNF : polynomial equation simplification normal form.
PQSDNF : polynomial equation simplification normal form.
RRADNFORM : Real root arbitrary domain normal form.
RRINFORM : Real root integral normal form.
SetDIPAGBVariableWeight : Set the DIPAGB variable weight list for the normal with sugar strategy.
SetPSugNormFunc : Set polynomial normal with sugar strategy normalform function in domain.
SetPSugSpolFunc : Set polynomial normal with sugar strategy S-polynomial function in domain.
SimplifyNf : simplify normal form.

normalform

ADEPNFJ	: Arbitrary domain extended polynomial normalform in the sense of Janet.
ADPNF	: Arbitrary domain polynomial normalform.
ADPNFJ	: Arbitrary domain polynomial normalform in the sense of Janet.
ADPNFJS	: Arbitrary domain polynomial normalform in the sense of Janet modulo a set
ADPSUGNF	: Arbitrary domain polynomial normal with sugar strategy normalform.
DILNFJ	: Distributive Polynomial List normalform in the sense of Janet.
DIPNF	: distributive polynomial normalform.
DIRRNF	: UGB distributive polynomial normalform.
NFORM	: Parametric normalform.
NFTOP	: Normalform by topreduction.
NSET	: Normalform set.
RDNORM	: Reduction normalform.
SetEDIPNormalform	: Set the extended distributive polynomial normalform function.
SetPNormFunc	: Set polynomial normalform function in domain.
SetPSugNormFunc	: Set polynomial normal with sugar strategy normalform function in domain.

normalforms

NFWRIT	: Normalforms write.
NWRIT0	: Normalforms write.
NWRIT1	: Normalforms write.
VRNORM	: Verify normalforms.

normalize

AFPNIP	: Algebraic number field polynomial normalize to integral polynomial.

normalized

DINTFE	: DIP normalized tupel field extension.
DINTSR	: DIP normalized tupel separation refinement.
DINTSS	: DIP normalized tupel strong separation.
DINTWR	: Distributive polynomial system normalized tupels write.

normalizing

RINT	: Rational interval normalizing transformation.

normative

NORMF	: Normative Factors.

not

DNOT	: Digit not.

null

IMUNS	: Integer matrix upper triangular matrix solution null space.
MMDNSB	: Matrix of modular digits null space basis.
NULRNV	: Rational number vector null test.
RNMUNS	: Rational number matrix upper triangular matrix solution null space.

number

ADDNFDIL : arbitrary domain domain number from distributive polynomial list.
ADDNFDILD : arbitrary domain domain number from distributive polynomial list
ADDNFDIP : arbitrary domain domain number from distributive polynomial.
ADDNFDIPD : arbitrary domain domain number from distributive polynomial or default.
ADDNFEDIP : Arbitrary domain domain number from extended distributive polynomial.
AFCOMP : Algebraic number field comparison.
AFDIF : Algebraic number field element difference.
AFFINT : Algebraic number field element from integer.
AFFRN : Algebraic number field element from rational number.
AFFRN : Algebraic number field element from rational number.
AFINV : Algebraic number field inverse.
AFNEG : Algebraic number field element negation.
AFPAFP : Algebraic number field polynomial algebraic number field element product.
AFPAFP : Algebraic number field polynomial algebraic number field element product.
AFPAFQ : Algebraic number field polynomial algebraic number field element quotient.
AFPAFQ : Algebraic number field polynomial algebraic number field element quotient.
AFPBRI : Algebraic number field polynomial basis real root isolation.
AFPCLL : Algebraic number field polynomial real root isolation, Collins-Loos
AFPDIF : Algebraic number field polynomial difference.
AFPDMV : Algebraic number field polynomial derivative, main variable.
AFPEMV : Algebraic number field polynomial evaluation of main variable.
AFPEV : Algebraic number field polynomial evaluation.
AFPFIP : Algebraic number field polynomial from integral polynomial.
AFPFRP : Algebraic number field polynomial from rational polynomial.
AFPINT : Algebraic number field polynomial integration.
AFPME : Algebraic number field, polynomial multiple evaluation.
AFPMON : Algebraic number field polynomial monic.
AFPMPR : Algebraic number field polynomial minimal polynomial of a real root.
AFPNEG : Algebraic number field polynomial negative.
AFPNIP : Algebraic number field polynomial normalize to integral polynomial.
AFPPR : Algebraic number field polynomial product.
AFPQR : Algebraic number field polynomial quotient and remainder.
AFPRCL : Algebraic number field polynomial real root isolation, Collins-Loos algorithm.
AFPRII : Algebraic number field polynomial real root isolation induction.
AFPRLS : Algebraic number field polynomial real root list separation.
AFPROD : Algebraic number field element product.
AFPRRI : Algebraic number field polynomial relative real root isolation.
AFPRRS : Algebraic number field polynomial real root separation.
AFPSUM : Algebraic number field polynomial sum.
AFQ : Algebraic number field quotient.
AFSIGN : Algebraic number field sign.
AFSUM : Algebraic number field element sum.
AFSUPB : Algebraic number field squarefree univariate polynomial squarefree
AFUPBA : Algebraic number field univariate polynomial squarefree basis

AFUPCB : Algebraic number field univariate polynomial coarsest squarefree basis.
AFUPGC : Algebraic number field univariate polynomial greatest common divisor
AFUPGS : Algebraic number field polynomial greatest squarefree divisor.
AFUPRB : Algebraic number field univariate polynomial root bound.
AFUPRL : Algebraic number field polynomial, root of a linear polynomial.
AFUPSF : Algebraic number field univariate polynomial squarefree factorization.
AFUPSR : Algebraic number field univariate polynomial, sign at a rational
ANDWR : Algebraic number decimal write.
ANFAF : Algebraic number from algebraic number field element.
ANFAF : Algebraic number from algebraic number field element.
ANIIPE : Algebraic number isolating interval for a primitive element.
ANPEDE : Algebraic number primitive element for a double extension.
ANREPE : Algebraic number represent element of a primitive extension.
APFRN : Arbitrary precision floating point from rational number.
APNELD : Arbitrary precision floating point number of equal leading digits.
CABS : Complex number absolute value.
CCOMP : Complex number comparison.
CCON : Complex number conjugate.
CDIF : Complex number difference.
CDREAD : Complex number decimal read.
CDWRITE : Complex number decimal write.
CEXP : Complex number exponentiation.
CgbI : Comprehensive Groebner basis number of conditions part.
CIM : Complex number imaginary part.
CINT : Complex number from integer.
CNEG : Complex number negative.
CNINV : Complex number inverse.
CNREAD : Complex number read.
CNWRITE : Complex number write.
CONE : Complex number one.
CPROD : Complex number product.
CQ : Complex number quotient.
CRAND : Complex number, random.
CRE : Complex number real part.
CRN : Complex number from rational number.
CRN : Complex number from rational number.
CRNP : Complex number from pair of rational numbers.
CSUM : Complex number sum.
DIGBSI : Distributive polynomial system algebraic number G basis sign.
DIPNBC : Distributive polynomial number of base coefficients.
DIPNOV : Distributive polynomial number of variables.
DIPRNPOL : DIP Rational Number Polynomial Definition Module.
DIRPRP : Distributive rational polynomial rational number product.
DIRPRQ : Distributive rational polynomial rational number quotient.
DOMAF : MAS Domain Algebraic Number Definition Module.
DOMC : MAS Domain Complex Number Definition Module.
DomLoadAF : Domain load algebraic number.
DomLoadC : Domain load complex number.
DomLoadO : Domain load octonion number.
DomLoadQ : Domain load quaternion number.
DomLoadRN : Domain load rational number.
DOMO : MAS Domain Octonion Number Definition Module.

DOMQ : MAS Domain Quaternion Number Definition Module.
DOMRN : MAS Domain Rational Number Definition Module.
EVNNZE : Exponent vector number of non zero exponents.
FFRN : Floating point from rational number.
GSYNSP : G-Symmetric Number of Special Polynomials.
IMFRNM : Integer matrix from rational number matrix.
IMFRNM1 : Integer matrix from rational number matrix.
IPAFME : Integral polynomial, algebraic number field multiple evaluation.
IUPMRN : Integral univariate polynomial minimal polynomial of a rational number.
IVFRNV : Integer vector from rational number vector.
IVFRNV1 : Integer vector from rational number vector.
LINALGRN : MAS Linear Algebra Rational Number Definition Module.
MASC : MAS Complex Number Definition Module.
MASO : MAS Octonion Number Definition Module.
MASQ : MAS Quaternion Number Definition Module.
MASRN : MAS Rational Number Definition Module.
NOENSP : Noether Number of Noetherian Base Polynomials.
NULRNV : Rational number vector null test.
OABS : Octonion number absolute value.
OCOMP : Octonion number comparison.
OCON : Octonion number conjugate.
ODIF : Octonion number difference.
ODREAD : Octonion number decimal read.
ODWRITE : Octonion number decimal write.
OEXP : Octonion number exponentiation.
OIM : Octonion number imaginary part.
OINT : Octonion number from integer.
ONEG : Octonion number negative.
ONINV : Octonion number inverse.
ONREAD : Octonion number read.
ONWRITE : Octonion number write.
OONE : Octonion number one.
OPROD : Octonion number product.
OQ : Octonion number quotient.
ORAND : Octonion number, random.
ORE : Octonion number real part.
ORN : Octonion number from rational number.
ORN : Octonion number from rational number.
ORNP : Octonion number from pair of rational numbers.
OSUM : Octonion number sum.
QABS : Quaternion number absolute value.
QCOMP : Quaternion number comparison.
QCON : Quaternion number conjugate.
QDIF : Quaternion number difference.
QDREAD : Quaternion number decimal read.
QDWRITE : Quaternion number decimal write.
QEXP : Quaternion number exponentiation.
QIMi : Quaternion number imaginary part i.
QIMj : Quaternion number imaginary part j.
QIMk : Quaternion number imaginary part k.
QINT : Quaternion number from integer.
QINV : Quaternion number inverse.

QNEG : Quaternion number negative.
QNREAD : Quaternion number read.
QNWRITE : Quaternion number write.
QONE : Quaternion number one.
QPROD : Quaternion number product.
QQ : Quaternion number quotient.
QRAND : Quaternion number, random.
QRE : Quaternion number real part.
QRN : Quaternion number from rational number.
QRN : Quaternion number from rational number.
QRN4 : Quaternion number from 4-tuple of rational numbers.
QSUM : Quaternion number sum.
RFNOV : Rational function number of variables.
RIRNP : Rational interval rational number product.
RNABS : Rational number absolute value.
RNBCR : Rational number binary common representation.
RNCEIL : Rational number, ceiling of.
RNCOMP : Rational number comparison.
RNDEN : Rational number denominator.
RNDIF : Rational number difference.
RNDRD : Rational number decimal read.
RNDRD : Rational number decimal read.
RNDWR : Rational number decimal write.
RNDWR : Rational number decimal write.
RNDWRS : Rational number decimal write special.
RNEXP : Rational number exponentiation.
RNFAP : Rational number from arbitrary precision floating point.
RNFCL2 : Rational number floor and ceiling of logarithm, base 2.
RNFF : Rational number from floating point.
RNFLOR : Rational number, floor of.
RNINT : Rational number from integer.
RNINV : Rational number inverse.
RNMAX : Rational number maximum.
RNMDET : Rational number matrix determinant, using Gaussian elimination.
RNMDETL : Rational number matrix determinant, using Laplace expansion.
RNMDIF : Rational number matrix difference.
RNMFIM : Rational number matrix from integer matrix.
RNMGE : Rational number matrix Gaussian elimination.
RNMGELUD : Rational number matrix Gaussian elimination LU-decomposition.
RNMHILBERT : Rational number matrix Hilbert.
RNMINV : Rational number matrix inversion.
RNMINVI : Rational number matrix inversion, integer algorithm.
RNMMAX : Rational number matrix maximum norm.
RNMPROD : Rational number matrix product.
RNMREAD : Rational number matrix read.
RNMSDS : Rational number matrix solve decomposed system.
RNMSUM : Rational number matrix sum.
RNMUNS : Rational number matrix upper triangular matrix solution null space.
RNMWRITE : Rational number matrix write.
RNNEG : Rational number negative.
RNNUM : Rational number numerator.
RNONE : Rational number one.
RNP2 : Rational number power of 2.

RNPROD : Rational number product.
RNQ : Rational number quotient.
RNRAND : Rational number, random.
RNREAD : Rational number read.
RNRED : Rational number reduction to lowest terms.
RNSIGN : Rational number sign.
RNSMPROD : Rational number scalar and matrix product.
RNSUM : Rational number sum.
RNSVPROD : Rational number vector product with scalar.
RNUM : Rational number unit matrix.
RNVABS : Rational number list absolute values.
RNVDIF : Rational number vector difference.
RNVFIV : Rational number vector from integer vector.
RNVLC : Rational number vector linear combination.
RNVMAX : Rational number vector maximum norm.
RNVQ : Rational number vector quotient.
RNVQF : Rational number vector quotient.
RNVREAD : Rational number vector read.
RNVSPROD : Rational number vector scalar product.
RNVSSUM : Rational number vector scalar sum.
RNVSVPROD : Rational number vector scalar vector product.
RNVSVSUM : Rational number vector scalar sum.
RNVVPROD : Rational number vector vector product.
RNVVSUM : Rational number vector vector sum.
RNVWRITE : Rational number vector write.
RNWRIT : Rational number write.
RPAFME : Rational polynomial, algebraic number field multiple evaluation.
RPRNP : Rational polynomial rational number product.
RUPMRN : Rational univariate polynomial minimal polynomial of a rational number.
SACANF : SAC Algebraic Number Field Definition Module.
SACPRIM : SAC Factorization and Prime Number Definition Module.
SACRN : SAC Rational Number Definition Module.
SCMULT : UGB rational exponent vector rational number product.
SEENR : UGB number of option.

numbers

CRNP : Complex number from pair of rational numbers.
DIPRNIB : DIP Rational Numbers Polynomial Definition Module in the sense of Janet.
LRNBMS : List of rational numbers bubble-merge sort.
LRNBS : List of rational numbers bubble sort.
LRNM : List of rational numbers merge.
LRNWRIT : List of rational numbers write.
ORNP : Octonion number from pair of rational numbers.
QRN4 : Quaternion number from 4-tuple of rational numbers.

numerator

RFNUM : Rational function numerator.
RNNUM : Rational number numerator.

object

COPYOB : Copy object.
DECOFTAG : Descriptor of tagged object.
OREAD : Object read.
OWRITE : Object write.
TAG : Tag object.
TYPOFTAG : Type of tagged object.
VALOFTAG : Value of tagged object.

occurs

OCCURQ : Occurs test equal pointers.

octonion

DomLoadO : Domain load octonion number.
DOMO : MAS Domain Octonion Number Definition Module.
MASO : MAS Octonion Number Definition Module.
OABS : Octonion number absolute value.
OCOMP : Octonion number comparison.
OCON : Octonion number conjugate.
ODIF : Octonion number difference.
ODREAD : Octonion number decimal read.
ODWRITE : Octonion number decimal write.
OEXP : Octonion number exponentiation.
OIM : Octonion number imaginary part.
OINT : Octonion number from integer.
ONEG : Octonion number negative.
ONINV : Octonion number inverse.
ONREAD : Octonion number read.
ONWRITE : Octonion number write.
OONE : Octonion number one.
OPROD : Octonion number product.
OQ : Octonion number quotient.
ORAND : Octonion number, random.
ORE : Octonion number real part.
ORN : Octonion number from rational number.
ORNP : Octonion number from pair of rational numbers.
OSUM : Octonion number sum.

odd

IODD : Integer odd.
MASODD : Odd.

off

MVOFF : modula variable off.

old

longjmp : Long jump to old environment.
SYONP : Syzygy for old Polynomials by new Polynomials.

one

ADONE : Arbitrary domain one.
CONE : Complex number one.
DIIPON : Distributive integral polynomial one.
DIPONE : distributive polynomial arbitrary domain one.
DIRPON : Distributive rational polynomial one.
FFONE : Finite field one.
IPONE : Integral polynomial one.
NLSYONP : Syzygy for one Polynomial.
OONE : Octonion number one.
PROJ : UGB projection, one dimension.
QONE : Quaternion number one.
RFONE : Rational function one.
RNONE : Rational number one.
RPONE : Rational polynomial one.
SetOneFunc : Set one test function in domain.

open

IPRIMO : Integral polynomial real root isolation, modified Uspensky method, open interval.
IUPVOI : Integral univariate polynomial, variations for open interval.
masReadOpen : MAS Read Open

operation

FORELIMXOPS : formula eliminate extended operation symbols.
FORGOP : formula get operation.
FORMKBINOP : formula make binary operation.
FORMKUNOP : formula make unary operation.
FORPBINOP : formula parse binary operation.
FORPBINOPA : formula parse binary operation argument.
FORPUNOP : formula parse unary operation.
FORPUNOPA : formula parse unary operation argument.
PQELIMXOPS : polynomial equation eliminate extended operation symbols.
PQELIMXOPS1 : polynomial equation eliminate extended operation symbols.

optimisation

DIPVOP : Distributive polynomial variable ordering optimisation.

optimization

ADFACTO : Arbitrary domain factorization with variable order optimization.
DIPTOO : DIP Termorder Optimization Definition Module.
DIPVOPP : Distributive polynomial variable ordering optimization and permutation
SetFactoFunc : Set factorization with variable order optimization function in domain.

option

RQEOPTWR	: real quantifier elimination option write.
SEENR	: Find key for option.
SEENR	: UGB number of option.
SetDCIBVarOrdOpt	: Set decompositional involutive base variable order option.
SetDIPAGBStrategy	: Set the DIPAGB strategy option for the extended critical pair selection.
WriteDIPAGBStrategy	: Write the DIPAGB strategy option for the extended critical pair selection

options

CGBOPT	: Comprehensive Groebner Basis Options.
CGBOPTWRITE	: Comprehensive Groebner Basis Options Write
EXECRD	: UGB execution options read.
OPREAD	: UGB options and parameter read.
PQOPT	: polynomial equation options.
PQOPTWR	: polynomial options write.
RQEOPTSET	: real quantifier elimination options set.
SetDCGBopt	: Set options for decompositional groebner bases.
SetDCIBopt	: Set decompositional involutive base options.
SetDIPIBopt	: Set distributive polynomial involutive base options.
WriteDCGBopt	: write decompositional groebner bases options.

or

ADDDFDIPD	: arbitrary domain domain descriptor from distributive polynomial or default.
ADDNFDIPD	: arbitrary domain domain number from distributive polynomial or default.
CallCompiled	: Call compiled function or procedure.
DMIA	: Define model, implementation or axioms.
DOR	: Digit or.
PQCRELAND	: contract relations or.
PQCRELOR	: contract relations or.
Signature	: Signature of a compiled function or procedure.

orbit

GINOPL	: G-Symmetric Integral Orbit Polynomial List.
GINORP	: G-Symmetric Integral Orbit Polynomial.
GRNOPL	: G-Symmetric Rational Orbit Polynomial List.
GRNORP	: G-Symmetric Rational Orbit Polynomial.
SUBOPL	: G-symmetric Orbit Polynomial List.
SUBORP	: Substitution Group Orbit Polynomial.

order

ADFACTO : Arbitrary domain factorization with variable order optimization.
DILIMO : distributive polynomial list inverse monomial order.
DIPIMO : distributive polynomial inverse monomial order.
DIPTODEF : DIP define distributive polynomial term order.
EVOWRITE : Exponent vector order write.
GSYORD : G-Symmetric Permutation Group Order.
IORD2 : Integer, order of 2.
MASORD : MAS order.
MASORDI : MAS order integer.
ORDER : Order.
PDBORD : Polynomial divided by order.
PORD : Polynomial order.
SetDCIBVarOrdOpt : Set decompositional involutive base variable order option.
SetFactoFunc : Set factorization with variable order optimization function in domain.
SUBORD : Substitution Group Order.

ordered

DIGBMI : Distributive minimal ordered groebner basis.
DIGMIN : Distributive minimal ordered groebner basis.
DIIFMI : Distributive minimal ordered groebner basis.
DIIGMI : Distributive minimal ordered groebner basis.
DINLGM : Distributive non-commutative minimal ordered left Groebner base.

ordering

DIPVOP : Distributive polynomial variable ordering optimisation.
DIPVOPP : Distributive polynomial variable ordering optimization and permutation

ore

OREC : Ore Condition.

out

CLOUT : Character list out.

output

CGBOUT : Comprehensive-Groebner-Basis execute and output.
DIPUV : Distributive polynomial univariate variable output.
dummyfactorize : LIST output.
IUPBEI : Integral univariate polynomial binary rational evaluation, integer output.
MLOGIO : Maslog Input Output System Definition Module.
MWRIT1 : Output in modula like syntax.
MWRITE : Output in modula like syntax.
OStreamKind : Output stream kind.
OUT : Output.
SOLINE : Set output line.
SOUNIT : Set output unit.
UGBBIN : UGB input, execute and output.
WriteDIPAGBTraceFlag : Write the DIPAGB trace flag in the output stream.
WriteDIPAGBVariableWeight : Write the DIPAGB variable weight list in the output stream.

over

DILFDILP : distributive polynomial list from distributive polynomial list over
DILPFDIL : distributive polynomials over polynomial ring list from distributive
DIPFDIPP : distributive polynomial from distributive polynomial over polynomial ring.
DIPPFDIP : distributive polynomial over polynomial ring from distributive polynomial.
DIRPDA : DIP rational polynomial ideal primary ideal decomposition over Q(alpha).
MPPFMP : monomial of polynomial ring over polynomial ring from monomial of
RQEPRRC : This global variable holds information over the actual polynomial ring

p0

Compiledp0 : Compiled function declaration p0.

p1

Compiledp1 : Compiled function declaration p1.

p1v2

Compiledp1v2 : Compiled function declaration p1v2.

p1v3

Compiledp1v3 : Compiled function declaration p1v3.

p2

Compiledp2 : Compiled function declaration p2.

p2v2

Compiledp2v2 : Compiled function declaration p2v2.

p2v3

Compiledp2v3 : Compiled function declaration p2v3.

p3

Compiledp3 : Compiled function declaration p3.

p3v2

Compiledp3v2 : Compiled function declaration p3v2.

p3v3

Compiledp3v3 : Compiled function declaration p3v3.

p4

Compiledp4 : Compiled function declaration p4.

p5

Compiledp5 : Compiled function declaration p5.

pack

PACK : Pack character list.
PACK : Pack character list.

package

PQDEMO : Demonstration for this package.

pair

CPEXTEND : Critical pair extend.
CRNP : Complex number from pair of rational numbers.
DIDPLCPL4 : Distributive domain polynomial list construct pair list.
DIIPLCPL4 : Distributive integral polynomial list construct pair list.
DILCPL : Distributive polynomial list construct pair list.
DILUPL : Distributive polynomial list update pair list.
DNLCPL : distributive polynomial non-noetherian left construct pair list.
DNLUPL : distributive polynomial non-noetherian left update pair list.
DNRCPL : distributive polynomial non-noetherian right construct pair list.
DNRUPL : distributive polynomial non-noetherian right update pair list.
ECPINSERT : Extended critical pair insertion.
ECPLCMHT : Extended critical pair select the least common multiple of head terms.
ECPPOLY1 : Extended critical pair select the first extended distributive polynomial.
ECPPOLY2 : Extended critical pair select the second extended distributive
ECPSELECT : Select an extended critical pair from the extended critical pair list.
ECPSELECT : Select an extended critical pair from the extended critical pair list.
ECPSUGAR : Extended critical pair select sugar.
ECPUNEXTEND : Extended critical pair un-extend.
ECPWRITE : Extended critical pair write.
ForEachinRep : For each pair (n,e) in r apply function f.
NEXTPAIR : Next Pair of Polynomials.
ORNP : Octonion number from pair of rational numbers.
PAIR : Pair.
SetCPExtend : Set the critical pair extension function.
SetDIPAGBStrategy : Set the DIPAGB strategy option for the extended critical pair selection.
SetECPInsert : Set the extended critical pair insertion function.
SetECPSelect : Set the extended critical pair selection procedure.
SetECPWrite : Set the extended critical pair write procedure.
UPDATE : Update of extended ideal basis and extended critical pair list as required
WriteDIPAGBStrategy : Write the DIPAGB strategy option for the extended critical pair selection

pair-list

EVPLM : Exponent vector pair-list merge.
EVPLSO : Exponent vector pair-list sort.

pair-merge

DIPLPM : Distributive polynomial list pair-merge sort.

pairs

DIDPCPLMS1 : Distributive domain polynomial list construct pairs list merge sort.
DIDPUCPL1 : Distributive domain polynomial update constructed pairs list.
DIIPCPLMS1 : Distributive integral polynomial list construct pairs list merge sort.
DIIPUCPL1 : Distributive polynomial D-update constructed pairs list.
GS1 : UGB generate stack of sorted polynomials and critical pairs 1.
GS2 : UGB generate stack of sorted polynomials and critical pairs 2.
LECPUNEXTEND : List of extended critical pairs un-extend.
LECPWRITE : List of extended critical pairs write.
LPAIRS : list pairs.
MKNEWP : Make new pairs.
MKNEWP : UGB make new critical pairs.
MKPAIR : Make pairs.
MKPAIR : UGB make critical pairs for polynomial list.
PRSCOP : Pairs copy.
WPAIRS : Write pairs of polynomials.
WPLIST : Write polynomials and pairs.

parameter

GENPL : Generate parameter list.
NextParm : Next Parameter.
OPREAD : UGB options and parameter read.
RDPAR : UGB read parameter.

parametric

DIMEXE : Parametric dimension exec.
GBDIFF : Parametric difference.
GRED : Parametric reduction.
NFEXEC : Parametric ideal membership exec.
NFORM : Parametric normalform.
PdCons : Parametric dimension construct.
PdF : Parametric dimension formula and dimension list part.
PdParts : Parametric dimension parts.
PdVd : Parametric dimension variable list and domain descriptor part.
PdWrite : Parametric dimension write.
RQEPRRC : Real Quantifier Elimination with Parametric Real Root Count.
SPOL : Parametric spolynom.
WRQFN0 : Write quantifier free formula for parametric ideal membership.

parse

Aparse : Parse a set of ALDES-2 declarations and algorithms.
DoParse : Do parse.
FORPARGS : formula parse arguments.
FORPBINOP : formula parse binary operation.
FORPBINOPA : formula parse binary operation argument.
FORPFOR : formula parse formula.
FORPLVAR : formula parse list of variables.
FORPQUANT : formula parse quantifier.
FORPQUANTA : formula parse quantifier arguments.
FORPUNOP : formula parse unary operation.
FORPUNOPA : formula parse unary operation argument.
FORPVAR : formula parse variable.
FORPVARA : formula parse variable arguments.
Parse : Parse program and generate code.
pqpaf : polynomial equation simplification parse atomic formula.
SwitchParse : Switch parsing between generic / non-generic parse.
tfpaf : type formula parse atomic formula.

parser

ALDPARSE : Aldes Parser Definition Module.
InitBbfParser : Initialize black-box formula parser.
MASPARSE : MAS Parser Definition Module.

parsing

SwitchParse : Switch parsing between generic / non-generic parse.

part

ADPCP : Arbitrary Domain polynomial content and primitive part.
ADPCPP : Arbitrary domain polynomial content and primitive part.
CdpCd : Case distinction and polynomial set case distinction part.
CdpPs : Case distinction and polynomial set polynomial set part.
CgbI : Comprehensive Groebner basis number of conditions part.
CgbP : Comprehensive Groebner basis polynomial list part.
CIM : Complex number imaginary part.
ColpCol : Coloured polynomial colouring part.
ColpPol : Coloured polynomial polynomial part.
CondNzero : Condition non-zero part.
CondPRead : Condition part read.
CondZero : Condition zero part.
CPART : Condition part.
CRE : Complex number real part.
DIILPP : Distributive integral polynomial list primitive part.
DIIPCP : Distributive integral polynomial content and primitive part.
DIPCPP : distributive polynomial content and primitive part.
DIPPCPP : distributive polynomial pseudo content and primitive part.
EIVAPP : Exterior integral vector absolute primitive part.
EIVCPP : Exterior integral vector content and primitive part.
EIVPP : Exterior integral vector primitive part.
FdD : Formula and dimension formula part.
FdF : Formula and dimension formula part.

FdV : Formula and dimension variable list part.
GsS : Groebner system system part.
IPCPP : Integral polynomial content and primitive part.
IPICPP : Integral polynomial integer content and primitive part.
IPIPP : Integral polynomial integer primitive part.
IPPP : Integral polynomial primitive part.
IPSCPP : Integral polynomial sign, content, and primitive part.
MPUCPP : Modular polynomial univariate content and primitive part.
MPUPP : Modular polynomial univariate primitive part.
OIM : Octonion number imaginary part.
ORE : Octonion number real part.
PdF : Parametric dimension formula and dimension list part.
PdVd : Parametric dimension variable list and domain descriptor part.
QIMi : Quaternion number imaginary part i.
QIMj : Quaternion number imaginary part j.
QIMk : Quaternion number imaginary part k.
QRE : Quaternion number real part.
SetPCppFunc : Set Content and primitive part function.
VdD : Variable list and domain descriptor domain descriptor part.
VdV : Variable list and domain descriptor variable list part.
WUPD : White part update.

partial

DIMPAD : distributive monomial partial derivation.
DIPPAD : distributive polynomial partial derivation.
DPCC : Digit partial cosequence calculation.
IBCPS : Integer binomial coefficient partial sum.

partition

CSFPAR : Characteristic set from partition.
PARTN : Partition, next.
PARTR : Partition, random.
PARTSS : Partition sumset.

parts

CdpParts : Case distinction and polynomial set parts.
CgbParts : Comprehensive Groebner basis parts.
ColParts : Colouring parts.
ColpParts : Coloured polynomial parts.
CondParts : Condition parts.
FdParts : Formula and dimension parts.
GsParts : Groebner system parts.
IPLCPP : Integral polynomial list of contents and primitive parts.
PdParts : Parametric dimension parts.
VdParts : Variable list and domain descriptor parts.

pattern

PatternAStart : pattern and start.

permutation

DILPERM : distributive polynomial list permutation of variables.
DIPERM : Distributive polynomial permutation of variables.
DIPVOPP : Distributive polynomial variable ordering optimization and permutation
GSYORD : G-Symmetric Permutation Group Order.
GSYPGR : G-Symmetric Permutation Group Read.
GSYPGW : G-Symmetric Permutation Group Write.
GSYSPG : Symmetric Permutation Group.
INVPERM : inverse permutation.
INVPERM : inverse permutation.
PERMCY : Permutation, cyclic.
PERMR : Permutation, random.
PPERMV : Polynomial permutation of variables.
PVDEMA : Permutation vector for degree matrix.
PVFLISTS : permutation vector from lists.

permute

LPERM : List permute.

pi

APPI : Arbitrary precision floating point pi.

PL

EX0PL : Exclude 0 from PL.

plynomial

ADEPheadterm : Arbitrary domain extended plynomial head-term.

point

APABS : Arbitrary precision floating point absolute value.
APCMPR : Arbitrary precision floating point compare.
APCOMP : Arbitrary precision floating point composition.
APDIFF : Arbitrary precision floating point difference.
APDWR : Algebraic point, decimal write.
APEXP : Arbitrary precision floating point exponentiation.
APEXPT : Arbitrary precision floating point exponent.
APFINT : Arbitrary precision floating point from integer.
APFRN : Arbitrary precision floating point from rational number.
APLG10 : Arbitrary precision floating point logarithm base 10.
APMANT : Arbitrary precision floating point mantissa.
APNEG : Arbitrary precision floating point negative.
APNELD : Arbitrary precision floating point number of equal leading digits.
APPI : Arbitrary precision floating point pi.
APPROD : Arbitrary precision floating point product.
APQ : Arbitrary precision floating point quotient.
APROOT : Arbitrary precision floating point n-th root.
APSHFT : Arbitrary precision floating point shift.
APSIGN : Arbitrary precision floating point sign.

APSPRE : Arbitrary precision floating point set precision.
APSUM : Arbitrary precision floating point sum.
APWRIT : Arbitrary precision floating point write.
DOMAPF : MAS Domain Arbitrary Precision Floating Point Definition Module.
DomLoadAPF : Domain load arbitrary precision floating point.
FEXP : Floating point exponentiation.
FFGI : Floating point from gamma integer.
FFINT : Floating point from integer.
FFRN : Floating point from rational number.
FLOG10 : Floating point logarithm base 10.
IFF : Integer from floating point.
MASAPF : MAS Arbitrary Precision Floating Point Definition Module.
MASF : MAS Floating Point Definition Module.
RNFAP : Rational number from arbitrary precision floating point.
RNFF : Rational number from floating point.

pointer

ELEMP : Elementary Pointer.
PROCP : Procedure Pointer.

pointers

MEMQ : Membership test equal pointers.
OCCURQ : Occurs test equal pointers.

points

IPCEVP : Integral polynomial, choice of evaluation points.

polynom

DIIFSP : Distributive integral function polynom S-polynomial.
DILATDG : Distributive polynom list add total degree.
DILEP2P : Distributive polynom list extended polynom to polynom.
DILEP2P : Distributive polynom list extended polynom to polynom.
DILEP2P : Distributive polynom list extended polynom to polynom.
DILFPL : Distributive polynomial list from polynom list.
DIPIB3 : Distributive polynom involutive base.
DIPPGL3 : Distributive polynom prolongation list.
DIRPCOM : Distributive rational polynom complete system.
DIRPIB2 : Distributive rational polynom involutive basis.
DVREAD : Polynom descriptor read.
KREISP : Kreisteilungs polynom.
PLFDIL : Polynomial list from distributive polynom list.

polynomia

pqtexwaf : polynomia equation tex write atomic formula.

polynomial

AD2DIP : arbitrary domain to distributive polynomial.
ADCAN : Arbitrary Polynomial Cancel.
ADCHARPOL : Arbitrary domain characteristic polynomial.
ADDDFDIL : arbitrary domain domain descriptor from distributive polynomial list.
ADDDFDILD : arbitrary domain domain descriptor from distributive polynomial list
ADDDFDIP : arbitrary domain domain descriptor from distributive polynomial.
ADDDFDIPD : arbitrary domain domain descriptor from distributive polynomial or default.
ADDLAST : Add last Polynomial.
ADDNFDIL : arbitrary domain domain number from distributive polynomial list.
ADDNFDILD : arbitrary domain domain number from distributive polynomial list
ADDNFDIP : arbitrary domain domain number from distributive polynomial.
ADDNFDIPD : arbitrary domain domain number from distributive polynomial or default.
ADDNFEDIP : Arbitrary domain domain number from extended distributive polynomial.
ADDPPOS : Add Polynomial P at Position.
ADEPCompose : Arbitrary domain extended polynomial compose.
ADEPCrumble : Arbitrary domain extended polynomial crumble.
ADEPdegree : Arbitrary domain extended polynomial degree.
ADEPleadingterm : Arbitrary polynomial leading term.
ADEPmark : Arbitrary domain extended polynomial mark.
ADEPNFJ : Arbitrary domain extended polynomial normalform in the sense of Janet.
ADEPpolynomial : Arbitrary domain extended polynomial polynomial.
ADEPpolynomial : Arbitrary domain extended polynomial polynomial.
ADEPtriple : Arbitrary domain extende polynomial triple.
ADEPuntriple : Arbitrary domain extended polynomial untriple.
ADFIP : Arbitrary domain from integral polynomial.
ADGJredI : Arbitrary domain polynomial G Janet-reducible modulo I.
ADIredG : Arbitrary domain polynomial set I reducible modulo G.
ADLGeqH : Arbitrary domain polynomial list G equal H.
ADLGinH : Arbitrary domain polynomial list G in H.
ADPCP : Arbitrary Domain polynomial content and primitive part.
ADPCPP : Arbitrary domain polynomial content and primitive part.
ADPFACT : Arbitrary domain polynomial factorization.
ADPFDIP : arbitrary domain polynomial from distributive polynomial.
ADPFDIP : arbitrary domain polynomial from distributive polynomial.
ADPFeqG : Arbitrary domain polynomial F equal G.
ADPIQ : Arbitrary domain polynomial integer quotient.
ADPNEG : Arbitrary domain polynomial negative.
ADPNF : Arbitrary domain polynomial normalform.
ADPNFJ : Arbitrary domain polynomial normalform in the sense of Janet.
ADPNFJS : Arbitrary domain polynomial normalform in the sense of Janet modulo a set
ADPSFF : Arbitrary domain polynomial squarefree factorization.
ADPSP : Arbitrary domain polynomial S-polynomial.
ADPSUGNF : Arbitrary domain polynomial normal with sugar strategy normalform.
ADPSUGSP : Arbitrary domain polynomial normal with sugar strategy S-polynomial.
ADTOIP : Arbitrary domain to integral polynomial conversion.

AFPAFP : Algebraic number field polynomial algebraic number field element product.
AFPAFQ : Algebraic number field polynomial algebraic number field element quotient.
AFPBRI : Algebraic number field polynomial basis real root isolation.
AFPCLL : Algebraic number field polynomial real root isolation, Collins-Loos
AFPDIF : Algebraic number field polynomial difference.
AFPDMV : Algebraic number field polynomial derivative, main variable.
AFPEMV : Algebraic number field polynomial evaluation of main variable.
AFPEV : Algebraic number field polynomial evaluation.
AFPFIP : Algebraic number field polynomial from integral polynomial.
AFPFIP : Algebraic number field polynomial from integral polynomial.
AFPFRP : Algebraic number field polynomial from rational polynomial.
AFPFRP : Algebraic number field polynomial from rational polynomial.
AFPINT : Algebraic number field polynomial integration.
AFPME : Algebraic number field, polynomial multiple evaluation.
AFPMON : Algebraic number field polynomial monic.
AFPMPR : Algebraic number field polynomial minimal polynomial of a real root.
AFPMPR : Algebraic number field polynomial minimal polynomial of a real root.
AFPNEG : Algebraic number field polynomial negative.
AFPNIP : Algebraic number field polynomial normalize to integral polynomial.
AFPNIP : Algebraic number field polynomial normalize to integral polynomial.
AFPPR : Algebraic number field polynomial product.
AFPQR : Algebraic number field polynomial quotient and remainder.
AFPRCL : Algebraic number field polynomial real root isolation, Collins-Loos algorithm.
AFPRII : Algebraic number field polynomial real root isolation induction.
AFPRLS : Algebraic number field polynomial real root list separation.
AFPRRI : Algebraic number field polynomial relative real root isolation.
AFPRRS : Algebraic number field polynomial real root separation.
AFPSUM : Algebraic number field polynomial sum.
AFSUPB : Algebraic number field squarefree univariate polynomial squarefree
AFUPBA : Algebraic number field univariate polynomial squarefree basis
AFUPCB : Algebraic number field univariate polynomial coarsest squarefree basis.
AFUPGC : Algebraic number field univariate polynomial greatest common divisor
AFUPGS : Algebraic number field polynomial greatest squarefree divisor.
AFUPRB : Algebraic number field univariate polynomial root bound.
AFUPRL : Algebraic number field polynomial, root of a linear polynomial.
AFUPRL : Algebraic number field polynomial, root of a linear polynomial.
AFUPSF : Algebraic number field univariate polynomial squarefree factorization.
AFUPSR : Algebraic number field univariate polynomial, sign at a rational
CdpCd : Case distinction and polynomial set case distinction part.
CdpCons : Case distinction and polynomial set construct.
CdpParts : Case distinction and polynomial set parts.
CdpPs : Case distinction and polynomial set polynomial set part.
CdpPs : Case distinction and polynomial set polynomial set part.
CdpRead : Case distinction and polynomial set read.
CdpVd : Case distinction and polynomial set variable list and domain descriptor
CdpWrite : Case distinction and polynomial set write.
CGBLM : CGB coloured distributive polynomial list merge.
CgbP : Comprehensive Groebner basis polynomial list part.

CHDEGL : Check degree of polynomial list.
ColpCol : Coloured polynomial colouring part.
ColpCons : Coloured polynomial construct.
ColpConsCond : Coloured polynomial construct from condition.
ColpHT : Coloured polynomial head term.
ColpIsCnst : Coloured polynomial is (non zero) constant.
ColpIsZero : Coloured polynomial is zero.
ColpParts : Coloured polynomial parts.
ColpPol : Coloured polynomial polynomial part.
ColpPol : Coloured polynomial polynomial part.
DETPOL : Determine polynomial.
DFP : UGB distributive rational polynomial difference.
DIDIMS : Distributive polynomial dimension maximal independent set.
DIDIMWR : Distributive polynomial dimension write.
DIDPALCMPC : Distributive domain polynomial array list check and mark polynomials.
DIDPCPLMS1 : Distributive domain polynomial list construct pairs list merge sort.
DIDPDGB : Distributive domain polynomial D-groebner basis.
DIDPDNF : Distributive domain polynomial D-normal form.
DIDPEGB : Distributive domain polynomial E-groebner basis.
DIDPELIMDGB : Distributive domain polynomial eliminate D-groebner base.
DIDPENF : Distributive domain polynomial E-normal form.
DIDPGPOL : Distributive domain polynomial g polynomial.
DIDPGPOL : Distributive domain polynomial g polynomial.
DIDPLCPL4 : Distributive domain polynomial list construct pair list.
DIDPLEXTAL : Distributive domain polynomial list extend array list.
DIDPLM1 : Distributive domain polynomial list merge sort.
DIDPREDDGB : Distributive domain polynomial reduce D-groebner base.
DIDPSPOL : Distributive domain polynomial s polynomial.
DIDPSPOL : Distributive domain polynomial s polynomial.
DIDPSPOL2 : Distributive domain polynomial s polynomial.
DIDPSPOL2 : Distributive domain polynomial s polynomial.
DIDPTDR : Distributive domain polynomial top-D-reduzibel.
DIDPUCPL1 : Distributive domain polynomial update constructed pairs list.
DIFIP : Distributive polynomial from distributive integral polynomial.
DIFIP : Distributive polynomial from distributive integral polynomial.
DIFPF : Distributive polynomial with arbitrary domain coefficients from
DIGBC3 : Distributive polynomial groebner basis criterion 3.
DIGBC4 : Distributive polynomial groebner basis criterion 4.
DIGBSI : Distributive polynomial system algebraic number G basis sign.
DIGBZT : Distributive polynomial groebner base common zero test.
DIIFGB : Distributive integral function polynomial groebner basis.
DIIFLS : Distributive integral function polynomial list irreducible set.
DIIFNF : Distributive integral function polynomial normal form.
DIIFRP : Distributive integral polynomial from rational polynomial.
DIIFRP : Distributive integral polynomial from rational polynomial.
DIIGBA : Distributive integral polynomial groebner basis augmentation.
DIILFR : Distributive integral polynomial list from rational polynomial list.
DIILFR : Distributive integral polynomial list from rational polynomial list.
DIILIS : Distributive integral polynomial list irreducible set.
DIILIS : Distributive integral polynomial list irreducible set.
DIILISJ : Distributive integral polynomial list irreducible set.
DIILPP : Distributive integral polynomial list primitive part.

DIILRD : Distributive integral polynomial list read.
DIILWR : Distributive integral polynomial list write.
DIIPAB : Distributive integral polynomial absolute value.
DIIPALCMPC : Distributive integral polynomial array list check and mark polynomials.
DIIPCOM : Distributive integral polynomial complete system.
DIIPCP : Distributive integral polynomial content and primitive part.
DIIPCPLMS1 : Distributive integral polynomial list construct pairs list merge sort.
DIIPDF : Distributive integral polynomial difference.
DIIPDGB : Distributive integral polynomial D-groebner basis.
DIIPDM : Distributive integral polynomial derivation main variable.
DIIPDNF : Distributive integral polynomial normal form.
DIIPDR : Distributive integral polynomial derivation.
DIIPEGB : Distributive integral polynomial E-groebner basis.
DIIPELIMDGB : Distributive integral polynomial eliminate D-groebner base.
DIIPEM : Distributive integral polynomial evaluation of main variable.
DIIPENF : Distributive integral polynomial e-normal form.
DIIPEV : Distributive integral polynomial evaluation of the i-th variable.
DIIPEX : Distributive integral polynomial exponentiation.
DIIPGB : Distributive integral polynomial groebner basis.
DIIPGPOL : Distributive integral polynomial g polynomial.
DIIPGPOL : Distributive integral polynomial g polynomial.
DIIPHD : Distributive integral polynomial higher derivation.
DIIPIB2 : Distributive integral polynomial involutive base.
DIIPIB3 : Distributive integral polynomial involutive base.
DIIPIP : Distributive integral polynomial integer product.
DIIPIQ : Distributive integral polynomial integer quotient.
DIIPLCPL4 : Distributive integral polynomial list construct pair list.
DIIPLEXTAL : Distributive integral polynomial list extend array list.
DIIPLM1 : Distributive integral polynomial list merge sort.
DIIPLS : Distributive integral polynomial list sum.
DIIPMN : Distributive integral polynomial maximum norm.
DIIPNF : Distributive integral polynomial normal form.
DIIPNFJ : Distributive integral polynomial normal form in the sense of Janet.
DIIPNG : Distributive integral polynomial negative.
DIIPNORM : Distributive integral polynomial norm.
DIIPON : Distributive integral polynomial one.
DIIPPR : Distributive integral polynomial product.
DIIPPR2 : Distributive integral polynomial product.
DIIPPS : Distributive integral polynomial pseudo-remainder.
DIIPQ : Distributive integral polynomial quotient.
DIIPQR : Distributive integral polynomial quotient and remainder.
DIIPRA : Distributive integral polynomial random.
DIIPRD : Distributive integral polynomial read.
DIIPREDDGB : Distributive integral polynomial reduce D-groebner base.
DIIPSG : Distributive integral polynomial sign.
DIIPSM : Distributive integral polynomial sum.
DIIPSN : Distributive integral polynomial sum norm.
DIIPSO : Distributive integral polynomial sort.
DIIPSP : Distributive integral polynomial s polynomial.
DIIPSP : Distributive integral polynomial s polynomial.
DIIPSPOL : Distributive integral polynomial s polynomial.
DIIPSPOL : Distributive integral polynomial s polynomial.

DIIPSPOL2 : Distributive integral polynomial s polynomial.
DIIPSPOL2 : Distributive integral polynomial s polynomial.
DIIPSU : Distributive integral polynomial substitution.
DIIPSV : Distributive integral polynomial substitution for main variable.
DIIPTDR : Distributive integral polynomial top-D-reduzibel.
DIIPTM : Distributive integral polynomial translation main variable.
DIIPTR : Distributive integral polynomial translation.
DIIPUCPL1 : Distributive polynomial D-update constructed pairs list.
DIIPWR : Distributive integral polynomial write.
DIIPWV : Distributive integral polynomial write with standard variable list.
DIIRAS : Distributive integral polynomial random sparse exponent vector.
DIITNT : Distributive polynomial system interval tupel from norm tupel.
DIITWR : Distributive polynomial system interval tupels write.
DILADNF : distributive polynomial list arbitrary domain normal form.
DILBSO : Distributive polynomial list bubble sort.
DILCAN : Distributive Polynomial Cancel.
DILCONV : distributive polynomial list conversion.
DILCPL : Distributive polynomial list construct pair list.
DILDIM : Distributive polynomial list dimension.
DILFDILP : distributive polynomial list from distributive polynomial list over
DILFDILP : distributive polynomial list from distributive polynomial list over
DILFEL : Distributive polynomial list from exponent vector list.
DILFPFL : Distributive polynomial list with arbitrary domain coefficients from
DILFPL : Distributive polynomial list from polynom list.
DILIMO : distributive polynomial list inverse monomial order.
DILINV : distributive polynomial list introduce new variable.
DILIS : Distributive polynomial list irreducible set.
DILISJ : Distributive polynomial list irreducible set in the sense of Janet.
DILISJ2 : Distributive polynomial list irreducible set.
DILMOC : distributive polynomial monic.
DILNFJ : Distributive Polynomial List normalform in the sense of Janet.
DILPERM : distributive polynomial list permutation of variables.
DILPFDIL : distributive polynomials over polynomial ring list from distributive
DILPROD : distributive polynomial list product.
DILRD : Distributive polynomial list read.
DILSUM : Distributive polynomial list sum.
DILTDG : Distributive polynomial list total degree
DILUPL : Distributive polynomial list update pair list.
DILWR : Distributive polynomial list write.
DINCCO : Distributive rational non-commutative polynomial commutator.
DINCCP : Distributive rational non-commutative polynomial center polynomial.
DINCCP : Distributive rational non-commutative polynomial center polynomial.
DINCCPpre : Distributive rational non-commutative polynomial center polynomial preparation.
DINCCPpre : Distributive rational non-commutative polynomial center polynomial preparation.
DINLIS : Distributive non-commutative polynomial list left irreducible set.
DINLMPG : Distributive non-commutative left rational minimal polynomial for a G basis.
DINLMPL : Distributive non-commutative left rational minimal polynomial list for a G basis.
DINLNF : Distributive non-commutative polynomial left normal form.
DINLRD : Distributive non-commutative polynomial list read.

DINLSP : Distributive non-commutative polynomial left S-polynomial.
DINNGB : DIP Groebner bases for non noetherian polynomial rings.
DINPEX : Distributive non-commutative polynomial exponentiation.
DINPPR : Distributive polynomial non-commutative product.
DINPQ : Distributive non-commutative Polynomial Quotient.
DINPRD : Distributive non-commutative polynomial read.
DINPTL : Distributive polynomial non-commutative product table lookup.
DINPTslT : Distributive polynomial non-commutative product table strict lex test.
DINPTU : Distributive polynomial non-commutative product table update.
DINTWR : Distributive polynomial system normalized tupels write.
DIP2AD : distributive polynomial to arbitrary domain.
DIP2SYM : Distributive polynomial to symbol term.
DIPADGB : distributive polynomial arbitrary domain groebner basis.
DIPADGBext : distributive polynomial arbitrary domain groebner basis extension.
DIPADGBRED : distributive polynomial groebner basis reduction.
DIPADGBunion : distributive polynomial arbitrary domain groebner basis union.
DIPADIRSET : distributive polynomial arbitrary domain irreducible set.
DIPADM : Distributive polynomial advance main variable.
DIPADNF : distributive polynomial arbitrary domain normal form.
DIPADS : Distributive polynomial advance and substitute.
DIPADV : Distributive polynomial advance.
DIPAGB : Distributive polynomial arbitrary domain Groebner basis.
DIPBCP : Distributive polynomial base coefficient product.
DIPBSO : Distributive polynomial bubble sort.
DIPC : DIP Common Polynomial System Definition Module.
DIPCJ : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPCLP : Distributiv Polynomial Class of Polynomial.
DIPCLP : Distributiv Polynomial Class of Polynomial.
DIPCLT : Distributiv Polynomial Class of Term.
DIPCMP : Distributive polynomial composition.
DIPCNST : distributive polynomial is constant.
DIPCNSTR : distributive polynomial constant relative to variables.
DIPCOM : Distributive polynomial complete.
DIPCONV : distributive polynomial conversion.
DIPCPP : distributive polynomial content and primitive part.
DIPCT : distributive polynomial coefficient tuple.
DIPDEG : Distributive polynomial degree.
DIPDEGI : distributive polynomial degree of i-th main variable.
DIPDEM : Distributive polynomial degree matrix.
DIPDEV : Distributive polynomial degree vector.
DIPDIF : Distributive polynomial difference.
DIPDPV : Distributive polynomial division by power of variable.
DIPEINFJ : Integral distributive extended polynomial normal form in the sense of Janet.
DIPENFJ : Distributive extended polynomial normal form in the sense of Janet.
DIPERM : Distributive polynomial permutation of variables.
DIPEVL : Distributive polynomial exponent vector leading monomial.
DIPEVP : Distributive polynomial exponent vector product.
DIPEXC : Distributive polynomial exchange variables.
DIPEXP : Distributive polynomial exponentiation.
DIPEXTEND : Distributive polynomial extension.
DIPFAC : distributive polynomial factorization.

DIPFADIP : distributive polynomial from arbitrary domain integral polynomial.
DIPFADIP : distributive polynomial from arbitrary domain integral polynomial.
DIPFDIPP : distributive polynomial from distributive polynomial over polynomial ring.
DIPFDIPP : distributive polynomial from distributive polynomial over polynomial ring.
DIPFDIPP : distributive polynomial from distributive polynomial over polynomial ring.
DIPFIP : distributive polynomial from integral polynomial.
DIPFIP : distributive polynomial from integral polynomial.
DIPFIRST : Distributive polynomial first polynomial,
DIPFIRST : Distributive polynomial first polynomial,
DIPFMO : Distributive polynomial from monomial.
DIPFP : Distributive polynomial from polynomial.
DIPFP : Distributive polynomial from polynomial.
DIPGB : Distributive polynomial groebner basis.
DIPIB : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPIB : Distributive polynomial involutive basis.
DIPIB2 : Distributive polynomial involutive basis.
DIPIB4 : Distributive polynomial involutive basis.
DIPIIB : DIP Integral Polynomial System Definition Module in the sense of Janet.
DIPIMO : distributive polynomial inverse monomial order.
DIPINFJ : Integral Distributive polynomial normal form in the sense of Janet.
DIPINFJS : Integral Distributive polynomial normal form in the sense of Janet modulo a
DIPINV : Distributive polynomial introduction of new variables.
DIPIPOL : DIP Integer Polynomial Definition Module.
DIPIRLJ : distributive polynomial interreduced list in the sense of Janet.
DIPIRLJ2 : Distributive polynomial list interreduced list in the sense of Janet.
DIPLBC : Distributive polynomial leading base coefficient.
DIPLDC : Distributive polynomial leading coefficient.
DIPLDM : Distributive polynomial list degree matrix.
DIPLDV : Distributive polynomial list dependency on variables.
DIPLFPFL : Distributive polynomial list from recursive polynomial
DIPLFPFL : Distributive polynomial list from recursive polynomial
DIPLIR : distributive polynomial list interreduce.
DIPLM : Distributive polynomial list merge.
DIPLMD : distributive polynomial list maximum degree.
DIPLPM : Distributive polynomial list pair-merge sort.
DIPLRS : Distributive polynomial list re-sort.
DIPMAD : Distributive polynomial monomial advance.
DIPMC2 : UGB distributive polynomial composition 2.
DIPMCP : Distributive polynomial monomial composition.
DIPMOC : Distributive polynomial monic.
DIPMPM : Distributive polynomial multiplication by power of main variable.
DIPMPV : Distributive polynomial multiplication by power of variable.
DIPMRD : Distributive polynomial monomial reductum.
DIPMST : Distributive polynomial monomial set.
DIPMVV : distributive polynomial minimal variable vector.
DIPNBC : Distributive polynomial number of base coefficients.
DIPNEG : Distributive polynomial negative.

DIPNF : distributive polynomial normalform.
DIPNFJ : Distributive polynomial normal form in the sense of Janet.
DIPNFJS : Distributive polynomial normal form in the sense of Janet modulo a set of
DIPNML : Distributive polynomial nonmultiple variable list.
DIPNOR : Distributive polynomial normal form.
DIPNOV : Distributive polynomial number of variables.
DIPONE : distributive polynomial arbitrary domain one.
DIPPAD : distributive polynomial partial derivation.
DIPPCPP : distributive polynomial pseudo content and primitive part.
DIPPFDIP : distributive polynomial over polynomial ring from distributive polynomial.
DIPPFDIP : distributive polynomial over polynomial ring from distributive polynomial.
DIPPFDIP : distributive polynomial over polynomial ring from distributive polynomial.
DIPPGL : Distributive polynomial prolongation list.
DIPPGL2 : Distributive polynomial prolongation list.
DIPPOWER : distributive polynomial power.
DIPQR : Distributive polynomial quotient and remainder.
DIPRED : Distributive polynomial reductum.
DIPRNIB : DIP Rational Numbers Polynomial Definition Module in the sense of Janet.
DIPRNPOL : DIP Rational Number Polynomial Definition Module.
DIPROD : Distributive polynomial product.
DIPRWTDG : Distributive polynomial rational-weighted total degree.
DIPS : distributive polynomial S-polynomial.
DIPSFF : distributive polynomial squarefree factorization.
DIPSP : Distributive polynomial S-polynomial.
DIPSSM : Distributive polynomial sort and select minimal.
DIPSUM : Distributive polynomial sum.
DIPTBC : Distributive polynomial trailing base coefficient.
DIPTCF : Distributive polynomial trailing coefficient.
DIPTCS : Distributive polynomial trailing coefficient specified variable.
DIPTDG : Distributive polynomial total degree.
DIPTODEF : DIP define distributive polynomial term order.
DIPTRM : Distributive polynomial terms.
DIPTYP : Distributive polynomial typ.
DIPUNT : Distributive polynomial univariate test.
DIPUV : Distributive polynomial univariate variable output.
DIPVDEF : DIP define distributive polynomial variable list.
DIPVL : Distributive Polynomial List of Variables.
DIPVOP : Distributive polynomial variable ordering optimisation.
DIPVOPP : Distributive polynomial variable ordering optimization and permutation
DIPXCM : distributive polynomial extract constant monomials.
DIREAD : Distributive polynomial read.
DIRFAC : Distributive rational polynomial factorisation.
DIRFIP : Distributive rational polynomial from integral polynomial.
DIRFIP : Distributive rational polynomial from integral polynomial.
DIRGBA : Distributive rational polynomial groebner basis augmentation.
DIRGBR : Distributive rational polynomial groebner basis recursion.
DIRLCT : Distributive rational polynomial list ideal containment test.

DIRLIP : Distributive rational polynomial list ideal product.
DIRLIS : Distributive rational polynomial list irreducible set.
DIRLISJ : Distributive rational polynomial list irreducible set.
DIRLPD : DIP rational polynomial ideal primary ideal decomposition.
DIRLPI : Distributive rational polynomial list primary ideal.
DIRLPW : DIP rational polynomial ideal primary ideal decomposition write.
DIRLRD : Distributive rational polynomial list read.
DIRLWR : Distributive rational polynomial list write.
DIRMPG : Distributive rational minimal polynomial for a groebner basis.
DIROWR : Distributive polynomial system real root write.
DIRPAB : Distributive rational polynomial absolute value.
DIRPDA : DIP rational polynomial ideal primary ideal decomposition over Q(alpha).
DIRPDF : Distributive rational polynomial difference.
DIRPDM : Distributive rational polynomial derivation main variable.
DIRPDR : Distributive rational polynomial derivation.
DIRPEM : Distributive rational polynomial evaluation of main variable.
DIRPES : Distributive rational polynomial elementary symmetric functions.
DIRPEV : Distributive rational polynomial evaluation of the i-th variable.
DIRPEX : Distributive rational polynomial exponentiation.
DIRPFT : Distributive rational polynomial from term.
DIRPHD : Distributive rational polynomial higher derivation.
DIRPLS : Distributive rational polynomial list sum.
DIRPMC : Distributive rational polynomial monic.
DIRPMN : Distributive rational polynomial maximum norm.
DIRPMV : Distributiv Polynomial multiplication with a variable.
DIRPNF : Distributive rational polynomial normal form.
DIRPNFJ : Distributive rational polynomial normal form in the sense of Janet.
DIRPNG : Distributive rational polynomial negative.
DIRPON : Distributive rational polynomial one.
DIRPPR : Distributive rational polynomial product.
DIRPQ : Distributive rational polynomial quotient.
DIRPQR : Distributive rational polynomial quotient and remainder.
DIRPRA : Distributive rational polynomial random.
DIRPRD : Distributive rational polynomial read.
DIRPRP : Distributive rational polynomial rational number product.
DIRPRQ : Distributive rational polynomial rational number quotient.
DIRPSE : Distributive rational polynomial symm.
DIRPSG : Distributive rational polynomial sign.
DIRPSM : Distributive rational polynomial sum.
DIRPSN : Distributive rational polynomial sum norm.
DIRPSO : Distributive rational polynomial sort.
DIRPSP : Distributive rational polynomial S polynomial.
DIRPSP : Distributive rational polynomial S polynomial.
DIRPSP : UGB distributive polynomial S-polynomial.
DIRPSR : Distributive rational polynomial symmetric function reduction.
DIRPSU : Distributive rational polynomial substitution.
DIRPSV : Distributive rational polynomial substitution for main variable.
DIRPTM : Distributive rational polynomial translation main variable.
DIRPTR : Distributive rational polynomial translation.
DIRPWR : Distributive rational polynomial write.
DIRPWV : Distributive rational polynomial write with standard variable
DIRRAS : Distributive rational polynomial, random sparse exponent vector.

DIRRNF : UGB distributive polynomial normalform.
DIWRIT : Distributive polynomial write.
DMPPRD : Dense modular polynomial product.
DMPSUM : Dense modular polynomial sum.
DMUPNR : Dense modular univariate polynomial natural remainder.
DNLCPL : distributive polynomial non-noetherian left construct pair list.
DNLUPL : distributive polynomial non-noetherian left update pair list.
DNRCPL : distributive polynomial non-noetherian right construct pair list.
DNRUPL : distributive polynomial non-noetherian right update pair list.
DOMIP : MAS Domain Integral Polynomial Definition Module.
DOMRP : MAS Domain Rational Polynomial Definition Module.
DPFP : Dense polynomial from polynomial.
DPFP : Dense polynomial from polynomial.
ECPPOLY1 : Extended critical pair select the first extended distributive polynomial.
EDIIFSUGNF : Extended distributive integral function polynomial normal with sugar
EDIIFSUGSP : Extended distributive integral function polynomial normal with sugar
EDIPEVL : Extended distributive polynomial exponent vector of the leading monomial.
EDIPNOR : Extended distributive polynomial normal form.
EDIPSP : Extended distributive polynomial S-polynomial.
EDIPSUGAR : Extended distributive polynomial sugar.
EDIPSUGCON : Extended distributive polynomial normal with sugar strategy constructor.
EDIPSUGNOR : Extended distributive polynomial normal with sugar strategy normal form.
EDIPSUGSP : Extended distributive polynomial normal with sugar strategy S-polynomial.
EDIPUNEXTEND : Extended distributive polynomial un-extend.
EDIPWRITE : Extended distributive polynomial write.
EIVFUP : Exterior integral vector from univariate integral polynomial
EXPTU : UGB extract exponent vector list from polynomial list.
FILWRITE : Polynomial conversion.
FINDPI : Find polynomial.
FINDPITOP : Find polynomial.
GINCHK : G-Symmetric Integral Polynomial Check.
GINCUT : G-Symmetric Integral Polynomial Cut.
GINOPL : G-Symmetric Integral Orbit Polynomial List.
GINORP : G-Symmetric Integral Orbit Polynomial.
GINRED : G-Symmetric Integral Polynomial Reduction.
GRNCHK : G-Symmetric Rational Polynomial Check.
GRNCUT : G-Symmetric Rational Polynomial Cut.
GRNOPL : G-Symmetric Rational Orbit Polynomial List.
GRNORP : G-Symmetric Rational Orbit Polynomial.
GRNRED : G-Symmetric Rational Polynomial Reduction.
GSPREAD : G-symmetric polynomial read.
GSRREAD : G-symmetric rational polynomial read.
GSYINF : G-Symmetric Polynomial System Information.
GSYMFUIN : G-Symmetric Integral Polynomial System Definition Module.
GSYMFURN : G-Symmetric Rational Polynomial System Definition Module.
HDIFDI : Homogeneous distributive polynomial from distributive polynomial.
HDIFDI : Homogeneous distributive polynomial from distributive polynomial.
HIPRAN : Homogeneous integral polynomial random.

ICHARPOL : Integral matrix characteristic polynomial.
InitExternalsB : Initialize external compiled polynomial procedures.
InitExternalsC : Initialize external compiled non-commutative polynomial procedures.
IPABS : Integral polynomial absolute value.
IPAFME : Integral polynomial, algebraic number field multiple evaluation.
IPC : Integral polynomial content.
IPCEVP : Integral polynomial, choice of evaluation points.
IPCPP : Integral polynomial content and primitive part.
IPCRA : Integral polynomial chinese remainder algorithm.
IPCSFB : Integral polynomial coarsest squarefree basis.
IPDDADV : integral polynomial domain descriptor advance.
IPDDCMP : integral polynomial domain descriptor composition.
IPDECMP : integral polynomial domain element composition.
IPDER : Integral polynomial derivative.
IPDIF : Integral polynomial difference.
IPDMV : Integral polynomial derivative, main variable.
IPDSCR : Integral polynomial discriminant.
IPEMV : Integral polynomial evaluation of main variable.
IPEVAL : Integral polynomial evaluation.
IPEXP : Integral polynomial exponentiation.
IPFAC : Integral polynomial factorization.
IPFCB : Integral polynomial factor coefficient bound.
IPFLC : Integral polynomial factor list combine.
IPFLMER : integral polynomial factorlist merge.
IPFRP : Integral polynomial from rational polynomial.
IPFRP : Integral polynomial from rational polynomial.
IPFSD : Integral polynomial factorization, second derivative.
IPFSFB : Integral polynomial finest squarefree basis.
IPGCDC : Integral polynomial greatest common divisor and cofactors.
IPGFCB : Integral polynomial Gelfond factor coefficient bound.
IPGSUB : Integral polynomial general substitution.
IPHDMV : Integral polynomial higher derivative, main variable.
IPIC : Integral polynomial integer content.
IPICPP : Integral polynomial integer content and primitive part.
IPICS : Integral polynomial integer content subroutine.
IPIHOM : Integral polynomial mod ideal homomorphism.
IPINT : Integral polynomial integration.
IPIP : Integral polynomial integer product.
IPIPP : Integral polynomial integer primitive part.
IPIPR : Integral polynomial mod ideal product.
IPIQ : Integral polynomial integer quotient.
IPIQH : Integral polynomial mod ideal quadratic Hensel lemma.
IPLCM : Integral polynomial least common multiple.
IPLCPP : Integral polynomial list of contents and primitive parts.
IPLRRI : Integral polynomial list real root isolation.
IPMAXN : Integral polynomial maximum norm.
IPNEG : Integral polynomial negative.
IPONE : Integral polynomial one.
IPPGSD : Integral polynomial primitive greatest squarefree divisor.
IPPP : Integral polynomial primitive part.
IPPROD : Integral polynomial product.
IPPSC : Integral polynomial principal subresultant coefficients.
IPPSR : Integral polynomial pseudo-remainder.

IPQ : Integral polynomial quotient.
IPQR : Integral polynomial quotient and remainder.
IPRAN : Integral polynomial random.
IPRAN : Integral polynomial, random.
IPRCH : Integral polynomial real root calculation, high precision.
IPRCHS : Integral polynomial real root calculation, high-precision special.
IPRCN1 : Integral polynomial real root calculation, 1 root.
IPRCNP : Integral polynomial real root calculation, newton method preparation.
IPREAD : Integral polynomial read.
IPRES : Integral polynomial resultant.
IPRICL : Integral polynomial real root isolation, Collins-Loos algorithm.
IPRIM : Integral polynomial real root isolation, modified Uspensky method.
IPRIMO : Integral polynomial real root isolation, modified Uspensky method, open interval.
IPRIMS : Integral polynomial real root isolation, modified Uspensky method, standard interval.
IPRIMU : Integral polynomial real root isolation, modified Uspensky method, unit interval.
IPRIU : Integral polynomial real root isolation, Uspensky method.
IPRIUP : Integral polynomial real root isolation, Uspensky method, positive roots.
IPRPRS : Integral polynomial reduced polynomial remainder sequence.
IPRPRS : Integral polynomial reduced polynomial remainder sequence.
IPRRII : Integral polynomial real root isolation induction.
IPRRLS : Integral polynomial real root list separation.
IPRRRI : Integral polynomial relative real root isolation.
IPRRS : Integral polynomial real root separation.
IPSCPP : Integral polynomial sign, content, and primitive part.
IPSF : Integral polynomial squarefree factorization.
IPSFBA : Integral polynomial squarefree basis augmentation.
IPSFF : integral polynomial squarefree factorization
IPSIFI : Integral polynomial standard isolating interval from isolating interval.
IPSIGN : Integral polynomial sign.
IPSMV : Integral polynomial substitution for main variable.
IPSPRS : Integral polynomial subresultant polynomial remainder sequence.
IPSPRS : Integral polynomial subresultant polynomial remainder sequence.
IPSR : Integral polynomial specified roots.
IPSRM : Integral polynomial strong real root isolation, modified Uspensky method.
IPSRMS : Integral polynomial strong real root isolation, modified Uspensky method, standard interval.
IPSRP : Integral polynomial similiar to rational polynomial.
IPSRP : Integral polynomial similiar to rational polynomial.
IPSUB : Integral polynomial substitution.
IPSUM : Integral polynomial sum.
IPSUMN : Integral polynomial sum norm.
IPTPR : Integral polynomial truncated product.
IPTRAN : Integral polynomial translation.
IPTRMV : Integral polynomial translation, main variable.
IPTRUN : Integral polynomial truncation.
IPVCHT : Integral polynomial variations after circle to half-plane transformation.
IPWRIT : Integral polynomial write.

ISFPF : Integral squarefree polynomial factorization.
ISFPIR : Integral squarefree polynomial isolating interval refinement.
ISPSFB : Integral squarefree polynomial squarefree basis.
IUPBEI : Integral univariate polynomial binary rational evaluation, integer output.
IUPBES : Integral univariate polynomial binary rational evaluation of sign.
IUPBHT : Integral univariate polynomial binary homothetic transformation.
IUPBRE : Integral univariate polynomial binary rational evaluation.
IUPCHT : Integral univariate polynomial circle to half-plane transformation.
IUPFAC : Integral univariate polynomial factorization.
IUPFDS : Integral univariate polynomial factor degree set.
IUPIHT : Integral univariate polynomial integer homothetic transformation.
IUPMRN : Integral univariate polynomial minimal polynomial of a rational number.
IUPMRN : Integral univariate polynomial minimal polynomial of a rational number.
IUPNT : Integral univariate polynomial negative transformation.
IUPQH : Integral univariate polynomial quadratic Hensel lemma.
IUPQHL : Integral univariate polynomial quadratic Hensel lemma, list.
IUPRB : Integral univariate polynomial root bound.
IUPRC : Integral univariate polynomial resultant and cofactor.
IUPRLP : Integral univariate polynomial, root of a linear polynomial.
IUPRLP : Integral univariate polynomial, root of a linear polynomial.
IUPTPR : Integral univariate polynomial truncated product.
IUPTR : Integral univariate polynomial translation.
IUPTR1 : Integral univariate polynomial translation by 1.
IUPVAR : Integral univariate polynomial variations.
IUPVOI : Integral univariate polynomial, variations for open interval.
IUPVSI : Integral univariate polynomial, variations for standard interval.
IUSFPF : Integral univariate squarefree polynomial factorization.
LDEG : Distributive polynomial list total degree.
MASPGCD : MAS Polynomial GCD and RES System Definition Module.
MIPDIF : Modular integral polynomial difference.
MIPFSM : Modular integral polynomial from symmetric modular.
MIPHOM : Modular integral polynomial homomorphism.
MIPIPR : Modular integral polynomial mod ideal product.
MIPISE : Modular integral polynomial mod ideal, solution of equation.
MIPNEG : Modular integral polynomial negation.
MIPPR : Modular integral polynomial product.
MIPRAN : Modular integral polynomial, random.
MIPSUM : Modular integral polynomial sum.
MIUPQR : Modular integral univariate polynomial quotient and remainder.
MIUPSE : Modular integral univariate polynomial, solution of equation.
MKPAIR : UGB make critical pairs for polynomial list.
MKPOL : Make polynomial without green monomials.
MLPQSMPL : Masload Polynomial Equation Simplify Definition Module.
MMPIQR : Modular monic polynomial mod ideal quotient and remainder.
MPDIF : Modular polynomial difference.
MPEMV : Modular polynomial evaluation of main variable.
MPEVAL : Modular polynomial evaluation.
MPEXP : Modular polynomial exponentiation.
MPGCDC : Modular polynomial greatest common divisor and cofactors.
MPHOM : Modular polynomial homomorphism.

MPINT : Modular polynomial interpolation.
MPIQH : Modular polynomial mod ideal, quadratic Hensel lemma.
MPIQHL : Modular polynomial mod ideal quadratic Hensel lemma, list.
MPIQHS : Modular polynomial mod ideal, quadratic Hensel lemma on a single variable.
MPMDP : Modular polynomial modular digit product.
MPMON : Modular polynomial monic.
MPNEG : Modular polynomial negative.
MPPFMP : monomial of polynomial ring over polynomial ring from monomial of
MPPFMP : monomial of polynomial ring over polynomial ring from monomial of
MPPROD : Modular polynomial product.
MPPSR : Modular polynomial pseudo-remainder.
MPQ : Modular polynomial quotient.
MPQR : Modular polynomial quotient and remainder.
MPRAN : Modular polynomial, random.
MPRES : Modular polynomial resultant.
MPSPRS : Modular polynomial subresultant polynomial remainder sequence.
MPSPRS : Modular polynomial subresultant polynomial remainder sequence.
MPSUM : Modular polynomial sum.
MPUC : Modular polynomial univariate content.
MPUCPP : Modular polynomial univariate content and primitive part.
MPUCS : Modular polynomial univariate content subroutine.
MPUP : Modular polynomial univariate product.
MPUPP : Modular polynomial univariate primitive part.
MPUQ : Modular polynomial univariate quotient.
MTPLH : Matrix to Polynomial List Horizontal.
MTPLV : Matrix to Polynomial List Vertical.
MUPBQP : Modular univariate polynomial Berlekamp q polynomials construction.
MUPDDF : Modular univariate polynomial distinct degree factorization.
MUPDER : Modular univariate polynomial derivative.
MUPEGC : Modular univariate polynomial extended greatest common divisor.
MUPFBL : Modular univariate polynomial factorization-Berlekamp algorithm.
MUPFS : Modular univariate polynomial factorization, special.
MUPGCD : Modular univariate polynomial greatest common divisor.
MUPHEG : Modular univariate polynomial half-extended greatest common divisor.
MUPRAN : Modular univariate polynomial, random.
MUPRC : Modular univariate polynomial resultant and cofactor.
MUPRES : Modular univariate polynomial resultant.
MUPSFF : Modular univariate polynomial squarefree factorization.
NLPLMULT : Non-Commutative Polynomial List Multiplication.
NLSYONP : Syzygy for one Polynomial.
NOEINF : Noether Polynomial System Information.
NOEL32 : Noether SK Polynomial Computation.
NOEPIP : Noether Polynomial Factor Multiplication.
NOEPPR : Noether Polynomial Product.
NOEPSM : Noether Polynomial Sum.
NOERED : Noether G-Symmetric Polynomial Computation.
NOESRT : Noether Polynomial Sort.
NOETHER : Noether Polynomial System Definition Module.
PBCLI : Polynomial base coefficients list.
PBIN : Polynomial binomial.
PCL : Polynomial coefficient list.

PCOMP : UGB distributive polynomial composition.
PCONST : Polynomial constant.
PDBORD : Polynomial divided by order.
PDEG : Polynomial degree.
PDEGSV : Polynomial degree, specified variable.
PDEGV : Polynomial degree vector.
PDPV : Polynomial division by power of variable.
PFDIP : Polynomial from distributive polynomial.
PFDIP : Polynomial from distributive polynomial.
PFDP : Polynomial from dense polynomial.
PFDP : Polynomial from dense polynomial.
PFIDNOR : Integral Polynomial D Normal Form.
PFIGB : Integral Polynomial Groebner Basis.
PFIGBA : Integral Polynomial Groebner Basis augmentation.
PFILDNOR : Integral Polynomial List D-Normal Form.
PFILDS : Integral polynomial list d-irreducible set.
PFILNOR : Integral Polynomial List Normal Form.
PFILS : Integral polynomial list irreducible set.
PFINOR : Integral Polynomial Normal Form.
PFLDIPL : Recursive polynomial list (with domain-descriptor) from distributive
PFWRITE : Integral polynomial write.
PINV : Polynomial introduction of new variables.
PLBCF : Polynomial leading base coefficient.
PLDCF : Polynomial leading coefficient.
PLFDIL : Polynomial list from distributive polynom list.
PLHTP : Polynomial List Horizontal To Polynomial.
PLHTP : Polynomial List Horizontal To Polynomial.
PLMULT : Polynomial List Multiplication.
PLVTM : Polynomial List Vertical To Matrix.
PLWR : Polynomial List Write.
PMDEG : Polynomial modified degree.
PMON : Polynomial monomial.
PMPMV : Polynomial multiplication by power of main variable.
PMPV : Polynomial multiplication by power of variable.
PMWR : Polynomial Matrix Write.
POL : Polynomial at Position.
PORD : Polynomial order.
POS : Position of Polynomial.
PPERMV : Polynomial permutation of variables.
pqatom : polynomial equation atomic formula.
PQBASE : Polynomial Equation Base Definition Module.
PQBASE : symbol to mark a polynomial equation special atomic formula.
PQCnfSimplify : polynomial equation cnf based simplification.
PQDnfSimplify : polynomial equation dnf based simplification.
PQELIMXOPS : polynomial equation eliminate extended operation symbols.
PQELIMXOPS1 : polynomial equation eliminate extended operation symbols.
pqgpol : polynomial equation get polynomial.
pqgpol : polynomial equation get polynomial.
pqgrel : polynomial equation get relation symbol.
PQIREAD : polynomial equation infix read.
pqmkaf : polynomial equation simplification make atomic formula.
PQMKCNF : polynomial equation make disjunctive normal form.
PQMKDNF : polynomial equation make disjunctive normal form.

PQMKPOS : polynomial equation make positive.
PQMKPRENEX : polynomial equation make prenex.
PQMKVD : polynomial equation make variable names disjoint.
PQOPT : polynomial equation options.
PQOPTWR : polynomial options write.
pqpaf : polynomial equation simplification parse atomic formula.
PQPPRT : polynomial equation pretty print.
PQPREAD : polynomial equation read.
PQPRING : polynomial equation polynomial ring.
PQPRING : polynomial equation polynomial ring.
PQPRINGWR : polynomial equation polynomial ring write.
PQPRINGWR : polynomial equation polynomial ring write.
pqprtaf : polynomial equation simplification print atomic formula.
pqreadaf : polynomial equation read atomic formula.
PQSCNF : polynomial equation simplification normal form.
PQSDNF : polynomial equation simplification normal form.
PQSIMPLIFY : polynomial equation simplify.
pqsimplifyaf : polynomial equation simplify atomic formula.
PQSIMPLIFYP : polynomial equation simplify.
pqsmart : polynomial equation atomic formula smart simplification.
PQSMPL : Polynomial Equation Simplification Definition Module.
PQSMPL : polynomial equation simplify.
PQTEXW : polynomial equation tex write.
PREAD : UGB polynomial read.
PRED : Polynomial reductum.
PRT : Polynomial reciprocal transformation.
PSDSV : Polynomial special decomposition, specified variable.
PTBCF : Polynomial trailing base coefficient.
PTERM : Polynomial terms.
PTYP : Polynomial typ.
PUFP : Polynomial, univariate, from polynomial.
PUFP : Polynomial, univariate, from polynomial.
PUNT : Polynomial univariate test.
RDSYS : Read polynomial systems.
REFIND : Reduction find polynomial.
RFDDFIPDD : rational function domain descriptor from integral polynomial domain
RFFIP : Rational function from integral polynomial.
RPABS : Rational polynomial absolute value.
RPAFME : Rational polynomial, algebraic number field multiple evaluation.
RPBLGS : Rational polynomial base coefficients least common multiple, greatest common divisor, and sign.
RPCONST : Rational polynomial constant.
RPDIF : Rational polynomial difference.
RPDMV : Rational polynomial derivative, main variable.
RPEMV : Rational polynomial evaluation, main variable.
RPEXP : Rational polynomial exponentiation.
RPFIP : Rational polynomial from integral polynomial.
RPFIP : Rational polynomial from integral polynomial.
RPIMV : Rational polynomial integration, main variable.
RPLWRS : Rational polynomial list write.
RPMAIP : Rational polynomial monic associate of integral polynomial.
RPMAIP : Rational polynomial monic associate of integral polynomial.
RPMON : Rational polynomial monic.

RPNEG : Rational polynomial negative.
RPONE : Rational polynomial one.
RPPROD : Rational polynomial product.
RPQR : Rational polynomial quotient and remainder.
RPREAD : Rational polynomial read.
RPRNP : Rational polynomial rational number product.
RPSIGN : Rational polynomial sign.
RPSUM : Rational polynomial sum.
RPWRIT : Rational polynomial write.
RPWRTS : Rational polynomial write.
RQEPRRC : This global variable holds information over the actual polynomial ring
RRADPOLMATRIX : Real root arbitrary domain polynomial matrix.
RRIPIQ : Real root integral polynomial integral quotient.
RRIPOLMATRIX : Real root integral polynomial matrix.
RRIPQSUM : Real root integral polynomial quotient sum.
RRUADPOLTOVEC : Real root univariate arbitrary domain polynomial to vector.
RRUIPOLTOVEC : Real root univariate integral polynomial to vector.
RUPEGC : Rational univariate polynomial extended greatest common divisor.
RUPGCD : Rational univariate polynomial greatest common divisor.
RUPHEG : Rational univariate polynomial half-extended greatest common divisor.
RUPLCM : Rational univariate polynomial least common multiple.
RUPMRN : Rational univariate polynomial minimal polynomial of a rational number.
RUPMRN : Rational univariate polynomial minimal polynomial of a rational number.
SACDPOL : SAC Dense Polynomial Definition Module.
SACIPOL : SAC Integer Polynomial System Definition Module.
SACMPOL : SAC Modular Polynomial Definition Module.
SACMUFAC : SAC Modular Univariate Polynomial Factorization Definition Module.
SACPFAC : SAC Polynomial Factorization Definition Module.
SACPGCD : SAC Polynomial GCD and RES System Definition Module.
SACPOL : SAC Polynomial System Definition Module.
SACROOT : SAC Polynomial Real Root Definition Module.
SACRPOL : SAC Rational Polynomial Definition Module.
SACUPFAC : SAC Univariate Polynomial Factorization Definition Module.
SetDIPExtend : Set the distributive polynomial extension function.
SetDIPIBCancel : Set distributive polynomial involutive base cancel.
SetDIPIBCrit : Set distributive polynomial involutive base critera.
SetDIPIBopt : Set distributive polynomial involutive base options.
SetDIPIBSelect : Set distributive polynomial involutive base select.
SetDIPIIBSelect : Set Distributive integral polynomial Select.
SetEDIPNormalform : Set the extended distributive polynomial normalform function.
SetEDIPUnExtend : Set the extended distributive polynomial un-extension function.
SetEDIPWrite : Set the extended distributive polynomial write procedure.
SetFIPolFunc : Set from integral polynomial function in domain.
SetPNormFunc : Set polynomial normalform function in domain.
SetPSpolFunc : Set polynomial S-polynomial function in domain.
SetPSqfrFunc : Set polynomial squarefree factorization function in domain.
SetPSugNormFunc : Set polynomial normal with sugar strategy normalform function in domain.
SetPSugSpolFunc : Set polynomial normal with sugar strategy S-polynomial function in domain.

SFP : UGB distributive rational polynomial sum.
SMFMIP : Symmetric modular from modular integral polynomial.
SUBCHK : G-Symmetric Polynomial Check.
SUBINF : Substitution Group Polynomial System Information.
SUBOPL : G-symmetric Orbit Polynomial List.
SUBORP : Substitution Group Orbit Polynomial.
SUBRED : Noether G-Symmetric Polynomial Computation for Substitution Groups.
SUBST : Substitution Group Polynomial System Definition Module.
SUBSYM : G-symmetric Polynomial Symmetric Check.
SYM2DIP : Symbol term to distributive polynomial.
TFDIRP : Term from distributive rational polynomial.
UIPSIL : Univariate integral polynomial symmetric product with exterior index list.
UIPSIV : Univariate integral polynomial symmetric product with exterior integral vector.
VMPIP : Vector of modular polynomial inner product.
XDIPFPF : Distributive polynomial from recursive polynomial (with domain-descriptor).
XDIPFPF : Distributive polynomial from recursive polynomial (with domain-descriptor).
XPFDIP : Recursive polynomial (with domain-descriptor) from distributive polynomial.
XPFDIP : Recursive polynomial (with domain-descriptor) from distributive polynomial.

polynomials

ADDCGB : Add polynomials to comprehensive-groebner-basis.
CGBCOL : Write coloured polynomials without green monomials.
DCLWR : Coloured polynomials list write.
DET : Determine list of polynomials.
DIDPALCMPC : Distributive domain polynomial array list check and mark polynomials.
DIIPALCMPC : Distributive integral polynomial array list check and mark polynomials.
DILFPFL : Groebner bases and related procedures for recursive integral polynomials.
DILPFDIL : distributive polynomials over polynomial ring list from distributive
DIN1GB : Distributive non-commutative polynomials Groebner base.
DINCGB : Distributive non-commutative polynomials Groebner base.
DINLGB : Distributive non-commutative polynomials left Groebner base.
DIPIRL : distributive polynomials interreduced list of polynomials.
DIPIRL : distributive polynomials interreduced list of polynomials.
DIPRLF : distributive polynomials reduce list of polynomials with factor.
DIPRLF : distributive polynomials reduce list of polynomials with factor.
DIPTOOLS : Distributive Polynomials Tools Definition Module.
DIREGB : Distributive rational polynomials extended groebner basis.
DIRPGB : Distributive rational polynomials groebner basis.
DNN2GB : distributive polynomials non-noetherian 2-sided Groebner base.
DNNLGB : distributive non-noetherian polynomials left Groebner base.
DNNRGB : distributive polynomials non-noetherian right Groebner base.
DomLoadIP : Domain load integral polynomials .

DomLoadRP : Domain load rational polynomials .
EQPLCL : Equal lists of coloured polynomials.
GLEXTP : Global extraneous polynomials remove.
GREPOL : Get polynomials without green monomials.
GroebnerBases1 : Distributive polynomials decompositional groebner bases 1.
GroebnerBases2 : Distributive polynomials decompositional groebner bases 2.
GS1 : UGB generate stack of sorted polynomials and critical pairs 1.
GS2 : UGB generate stack of sorted polynomials and critical pairs 2.
GSYNSP : G-Symmetric Number of Special Polynomials.
LDIPEXTEND : List of distributive polynomials extend.
LEDIPUNEXTEND : List of extended distributive polynomials un-extend.
LEDIPWRITE : List of extended distributive polynomials write.
MAIPDE : Matrix of integral polynomials determinant, exact division algorithm.
MAIPDM : Matrix of integral polynomials determinant, modular algorithm.
MAIPHM : Matrix of integral polynomials homomorphism.
MAIPP : Matrix of integral polynomials product.
MCPMV : Matrix of coefficients of polynomials, with respect to main variable.
MERGE : Noether Merge of Base Polynomials.
MINPP : Minimize polynomials list.
MMPDMA : Matrix of modular polynomials determinant, modular algorithm.
MMPEV : Matrix of modular polynomials evaluation.
MUPBQP : Modular univariate polynomial Berlekamp q polynomials construction.
NEWDIF : UGB exponent vector list difference from polynomials.
NEXTPAIR : Next Pair of Polynomials.
NOENSP : Noether Number of Noetherian Base Polynomials.
REXTP : Remove extraneous polynomials.
SYONP : Syzygy for old Polynomials by new Polynomials.
SYONP : Syzygy for old Polynomials by new Polynomials.
UIPRES : Univariate integral polynomials resultant.
UIPRS1 : Univariate integral polynomials resultant 1.
UPDPP : Update polynomials.
VERIFY : Verify conditions and polynomials.
VIPIIP : Vector of integral polynomials with vector of integers inner product.
WPAIRS : Write pairs of polynomials.
WPLIST : Write polynomials and pairs.

pop

EvordPop : evord pop.
ValisPop : valis pop.

portability

Portab : Portability Definition Module.

position

ADDPPOS : Add Polynomial P at Position.
INSPOSV : Insert Position Vector.
POL : Polynomial at Position.
POS : Position of Polynomial.

positive

FORMKPOS : formula make positive.
IPRIUP : Integral polynomial real root isolation, Uspensky method, positive roots.
MLDMKPOS : maslog demonstration make positive.
MLDMKPOS1 : maslog demonstration make positive 1.
PQMKPOS : polynomial equation make positive.

postion

GENPOSV : Generate Postion Vector.

power

DIPDPV : Distributive polynomial division by power of variable.
DIPMPM : Distributive polynomial multiplication by power of main variable.
DIPMPV : Distributive polynomial multiplication by power of variable.
DIPPOWER : distributive polynomial power.
IDP2 : Integer division by power of 2.
IMP2 : Integer multiplication by power of 2.
IPOWER : Integer power.
NOEPOW : Noether SK Power Sum Computation.
PDPV : Polynomial division by power of variable.
PMPMV : Polynomial multiplication by power of main variable.
PMPV : Polynomial multiplication by power of variable.
POWSEV : Power of variable symmetric product with exterior vector.
RNP2 : Rational number power of 2.
SUBPOW : Noether SK Power Sum Computation for Substitution Groups.

PQS-procedures

InitExternalsPQSMPL : initialize external compiled PQS-procedures.

precision

APABS : Arbitrary precision floating point absolute value.
APCMPR : Arbitrary precision floating point compare.
APCOMP : Arbitrary precision floating point composition.
APDIFF : Arbitrary precision floating point difference.
APEXP : Arbitrary precision floating point exponentiation.
APEXPT : Arbitrary precision floating point exponent.
APFINT : Arbitrary precision floating point from integer.
APFRN : Arbitrary precision floating point from rational number.
APLG10 : Arbitrary precision floating point logarithm base 10.
APMANT : Arbitrary precision floating point mantissa.
APNEG : Arbitrary precision floating point negative.
APNELD : Arbitrary precision floating point number of equal leading digits.
APPI : Arbitrary precision floating point pi.
APPROD : Arbitrary precision floating point product.
APQ : Arbitrary precision floating point quotient.
APROOT : Arbitrary precision floating point n-th root.
APSHFT : Arbitrary precision floating point shift.
APSIGN : Arbitrary precision floating point sign.
APSPRE : Arbitrary precision floating point set precision.

APSPRE : Arbitrary precision floating point set precision.
APSUM : Arbitrary precision floating point sum.
APWRIT : Arbitrary precision floating point write.
DOMAPF : MAS Domain Arbitrary Precision Floating Point Definition Module.
DomLoadAPF : Domain load arbitrary precision floating point.
IPRCH : Integral polynomial real root calculation, high precision.
MASAPF : MAS Arbitrary Precision Floating Point Definition Module.
RNFAP : Rational number from arbitrary precision floating point.

precomputation

PUG : Universal Groebner base using precomputation.

precomputed

PLF : UGB linear form with precomputed linear forms.
PUGB : Universal Groebner base with precomputed linear forms.

predicate

SetElementP : Set element predicate.
SetElementPQ : Set element predicate equal.

prefix

FORPREAD : formula prefix read.
MLDPREAD : maslog demonstration prefix read.

prenex

FORMKPRENEX : formula make prenex.
FORMKPRENEX1 : formula make prenex 1.
FORMKPRENEXI : formula make prenex.
MLDMKPRENEX : maslog demonstration make prenex.
PQMKPRENEX : polynomial equation make prenex.

preparation

DINCCPpre : Distributive rational non-commutative polynomial center polynomial preparation.
IPRCNP : Integral polynomial real root calculation, newton method preparation.

prepare

FORPREPQE : formula prepare quantifier elimination.
MLDPREPQE : maslog demonstration prepare quantifier elimination.

pretty

FORPPRT : formula pretty print.
FORPPVAR : formula pretty print variable.
MLDPPRT : maslog demonstration pretty print.
PQPPRT : polynomial equation pretty print.
TFPPRT : type formula pretty print.

primality

ISPT : Integer selfridge primality test.

primary

DIRLPD : DIP rational polynomial ideal primary ideal decomposition.
DIRLPI : Distributive rational polynomial list primary ideal.
DIRLPW : DIP rational polynomial ideal primary ideal decomposition write.
DIRPDA : DIP rational polynomial ideal primary ideal decomposition over Q(alpha).

prime

DPGEN : Digit prime generator.
GDPGEN : Gaussian digit prime generator.
ILPDS : Integer large prime divisor search.
IMPDS : Integer medium prime divisor search.
ISPD : Integer small prime divisors.
SACPRIM : SAC Factorization and Prime Number Definition Module.

primitive

ADPCP : Arbitrary Domain polynomial content and primitive part.
ADPCPP : Arbitrary domain polynomial content and primitive part.
ANIIPE : Algebraic number isolating interval for a primitive element.
ANPEDE : Algebraic number primitive element for a double extension.
ANREPE : Algebraic number represent element of a primitive extension.
DIILPP : Distributive integral polynomial list primitive part.
DIIPCP : Distributive integral polynomial content and primitive part.
DIPCPP : distributive polynomial content and primitive part.
DIPPCPP : distributive polynomial pseudo content and primitive part.
EIVAPP : Exterior integral vector absolute primitive part.
EIVCPP : Exterior integral vector content and primitive part.
EIVPP : Exterior integral vector primitive part.
IPCPP : Integral polynomial content and primitive part.
IPICPP : Integral polynomial integer content and primitive part.
IPIPP : Integral polynomial integer primitive part.
IPLCPP : Integral polynomial list of contents and primitive parts.
IPPGSD : Integral polynomial primitive greatest squarefree divisor.
IPPP : Integral polynomial primitive part.
IPSCPP : Integral polynomial sign, content, and primitive part.
MPUCPP : Modular polynomial univariate content and primitive part.
MPUPP : Modular polynomial univariate primitive part.
SetPCppFunc : Set Content and primitive part function.

principal

IPPSC : Integral polynomial principal subresultant coefficients.

print

ALLLF : UGB all linear forms from stack of projections and print.
FORPPLVAR : formula print lvar.
FORPPRT : formula pretty print.
FORPPVAR : formula pretty print variable.
MLDPPRT : maslog demonstration pretty print.
PQPPRT : polynomial equation pretty print.
pqprtaf : polynomial equation simplification print atomic formula.
TFPPRT : type formula pretty print.

problem

ASSPR : Assignment problem.

proc

DEFPROC : Define generic proc function.

procedure

CallCompiled : Call compiled function or procedure.
CompSummary : Compiled function and procedure summary.
MASBIOSU : Procedure declarations.
MASLISPU : Procedure declarations.
MASSYM : Procedure declarations.
MASYMDIP : Procedure declarations.
PROCP : Procedure Pointer.
SetBranchProc : Set Branch-Procedure for procedure GroebnerBases2:
SetECPSelect : Set the extended critical pair selection procedure.
SetECPWrite : Set the extended critical pair write procedure.
SetEDIPWrite : Set the extended distributive polynomial write procedure.
SetFacSugar : Set Factor-Sugar for procedure GroebnerBases1:
SetInit : Set the initialization procedure.
SetReduceExp : Set Reduce-Exponent for procedure GroebnerBases2:
SetUpdate : Set the update procedure.
SetUpdateVariableWeight : Set the DIPAGB variable weight update procedure.
Signature : Signature of a compiled function or procedure.

procedures

DILFPFL : Groebner bases and related procedures for recursive integral polynomials.
InitExternals : Initialize external compiled procedures.
InitExternalsA : Initialize external compiled arithmetic procedures.
InitExternalsB : Initialize external compiled polynomial procedures.
InitExternalsC : Initialize external compiled non-commutative polynomial procedures.
InitExternalsD : Initialize external compiled ideal decomposition and root procedures.
InitExternalsE : Initialize external compiled arbitrary domain procedures.
InitExternalsG : Tell Modula and LISP about external compiled procedures.
InitExternalsI : Initialize external compiled interface procedures.
InitExternalsJ : Initialize external compiled arithmetic procedures.
InitExternalsL : Initialize external compiled linear algebra procedures.
InitExternalsM : Initialize external compiled real root procedures.
InitExternalsML : Initialze external compiled logic procedures.
InitExternalsMLDEMO : Initialize externals maslog demonstration procedures.
InitExternalsQ : Initialize external compiled arithmetic Q procedures.
InitExternalsS : Tell Modula and LISP about external compiled procedures.
InitExternalsU : Initialize external compiled utility procedures.

processes

ClocK : ClocK returns milliseconds of the processes cpu-time.

processing

SACLIST : SAC List Processing Definition Module.

PROCF1

DIPDCGB : TYPE PROCF1 = PROCEDURE(LIST): LIST;

product

ADMPROD : Arbitrary domain matrix product.
ADMPTRACE : Arbitrary domain matrix product trace.
ADPROD : Arbitrary domain product.
ADSMPROD : Arbitrary domain scalar and matrix product.
ADVSPROD : Arbitrary domain vector scalar product.
ADVSVPROD : Arbitrary domain vector scalar vector product.
AFPAFP : Algebraic number field polynomial algebraic number field element product.
AFPPR : Algebraic number field polynomial product.
AFPROD : Algebraic number field element product.
APPROD : Arbitrary precision floating point product.
COLPRD : Colour product.
CP2 : UGB linear form product with rational exponent vector list 2.
CPLEXN : Cartesian product, lexicographically next.
CPROD : Complex number product.
CQ2 : UGB linear form product with rational exponent vector list.
DIIPIP : Distributive integral polynomial integer product.
DIIPPR : Distributive integral polynomial product.
DIIPPR2 : Distributive integral polynomial product.
DILPROD : distributive polynomial list product.
DINPPR : Distributive polynomial non-commutative product.
DINPTL : Distributive polynomial non-commutative product table lookup.
DINPTslT : Distributive polynomial non-commutative product table strict lex test.
DINPTU : Distributive polynomial non-commutative product table update.
DIPBCP : Distributive polynomial base coefficient product.
DIPEVP : Distributive polynomial exponent vector product.
DIPROD : Distributive polynomial product.
DIRLIP : Distributive rational polynomial list ideal product.
DIRPPR : Distributive rational polynomial product.
DIRPRP : Distributive rational polynomial rational number product.
DMPPRD : Dense modular polynomial product.
DPR : Digit product.
EIVEPR : Exterior integral vector exterior product.
EIVILP : Exterior integral vector inner left product.
EIVIP : Exterior integral vector integer product.
EIVIRP : Exterior integral vector inner right product.
EVSSPROD : Exponent vektor set sorted product.
FFPROD : Finite field product.
IDIPR2 : Integer digit inner product, length 2.
IDPR : Integer-digit product.
IKM : Integer vector component product.

ILEXPR : Index list exterior product.
ILILPR : Index list inner left product.
ILINPR : Index list inner product.
ILIRPR : Index list inner right product.
IMPROD : Integer matrix product.
IMPTRACE : Integral matrix product trace.
IMRTPROD : Integral matrix right tensor product.
IPIP : Integral polynomial integer product.
IPIPR : Integral polynomial mod ideal product.
IPPROD : Integral polynomial product.
IPROD : Integer product.
IPROD : Integer product.
IPRODK : Integer product, Karatsuba algorithm.
IPTPR : Integral polynomial truncated product.
ISMPROD : Integer scalar and matrix product.
IUPTPR : Integral univariate polynomial truncated product.
IVSPROD : Integer vector scalar product.
IVSVPROD : Integer vector scalar and vector product.
IVVPROD : Integer vector vectors product.
MAIPP : Matrix of integral polynomials product.
MASNC : MAS Non-commutative Product Definition Module.
MDPROD : Modular digit product.
MIPIPR : Modular integral polynomial mod ideal product.
MIPPR : Modular integral polynomial product.
MIPROD : Modular integer product.
MPMDP : Modular polynomial modular digit product.
MPPROD : Modular polynomial product.
MPUP : Modular polynomial univariate product.
NOEPPR : Noether Polynomial Product.
OPROD : Octonion number product.
POWSEV : Power of variable symmetric product with exterior vector.
QPROD : Quaternion number product.
RFPROD : Rational function product.
RIRNP : Rational interval rational number product.
RNMPROD : Rational number matrix product.
RNPROD : Rational number product.
RNSMPROD : Rational number scalar and matrix product.
RNSVPROD : Rational number vector product with scalar.
RNVSPROD : Rational number vector scalar product.
RNVSVPROD : Rational number vector scalar vector product.
RNVVPROD : Rational number vector vector product.
RPPROD : Rational polynomial product.
RPRNP : Rational polynomial rational number product.
SCMULT : UGB rational exponent vector rational number product.
SCPROD : UGB rational exponent vector scalar product.
SetProdFunc : Set product function in domain.
SKPRO2 : UGB rational exponent vector scalar product with integer ev.
UIPSIL : Univariate integral polynomial symmetric product with exterior index list.
UIPSIV : Univariate integral polynomial symmetric product with exterior integral vector.
VIPIIP : Vector of integral polynomials with vector of integers inner product.
VISPR : Vector of integers scalar product.
VMPIP : Vector of modular polynomial inner product.

program

DOS : Call DOS program.
Parse : Parse program and generate code.

programms

CGBMAIN : Comprehensive-Groebner-Bases Main Programms Definition Module.

programs

CGBMISC : Comprehensive-Groebner-Bases Miscellaneous Programs Definition Module.
SYZHLP : Syzygy Utility Programs Definition Module.
SYZMAIN : Syzygy Main Programs Definition Module.

projection

PROJ : UGB projection, one dimension.
PROJEC : UGB projection to dimension 1.

projections

ALLELN : UGB all linear forms from stack of projections.
ALLLF : UGB all linear forms from stack of projections and print.
LFALL : UGB all linear forms from stack of projections 1.

prolongation

DIPPGL : Distributive polynomial prolongation list.
DIPPGL2 : Distributive polynomial prolongation list.
DIPPGL3 : Distributive polynom prolongation list.

property

GET : Get property.
GET : Get property.
REMPRP : Remove property.
REMPRP : Remove property.

prune

FORSIMPLIFYP : formula simplify prune.

pseudo

DIPPCPP : distributive polynomial pseudo content and primitive part.

pseudo-remainder

DIIPPS : Distributive integral polynomial pseudo-remainder.
IPPSR : Integral polynomial pseudo-remainder.
MPPSR : Modular polynomial pseudo-remainder.

push

EvordPush : evord push.
ValisPush : valis push.

put

PUT : Put.
PUT : Put.

quadratic

IPIQH : Integral polynomial mod ideal quadratic Hensel lemma.
IUPQH : Integral univariate polynomial quadratic Hensel lemma.
IUPQHL : Integral univariate polynomial quadratic Hensel lemma, list.
MPIQH : Modular polynomial mod ideal, quadratic Hensel lemma.
MPIQHL : Modular polynomial mod ideal quadratic Hensel lemma, list.
MPIQHS : Modular polynomial mod ideal, quadratic Hensel lemma on a single variable.
RRADQUADFORM : Real root arbitrary domain quadratic form.
RRIQUADFORM : Real root integral quadratic form.
RRUADQUADFORM : Real root univariate arbitrary domain quadratic form.
RRUIQUADFORM : Real root univariate integral quadratic form.

quantifier

CGBQFF : Comprehensive Groebner basis quantifier free formula.
CPART : Comprehensive-Groebner-Basis quantifier free formula.
FORIMQB : formula innermost quantifier block.
FORMKQUANT : formula make quantifier.
FORPQUANT : formula parse quantifier.
FORPQUANTA : formula parse quantifier arguments.
FORPREPQE : formula prepare quantifier elimination.
MLDPREPQE : maslog demonstration prepare quantifier elimination.
RQEOPTSET : real quantifier elimination options set.
RQEOPTWR : real quantifier elimination option write.
RQEPRRC : Real Quantifier Elimination with Parametric Real Root Count.
RQEQE : real quantifier elimination quantifier elimination.
RQEQE : real quantifier elimination quantifier elimination.
WRQFN0 : Write quantifier free formula for parametric ideal membership.

quaternion

DomLoadQ : Domain load quaternion number.
DOMQ : MAS Domain Quaternion Number Definition Module.
MASQ : MAS Quaternion Number Definition Module.
QABS : Quaternion number absolute value.
QCOMP : Quaternion number comparison.
QCON : Quaternion number conjugate.
QDIF : Quaternion number difference.
QDREAD : Quaternion number decimal read.
QDWRITE : Quaternion number decimal write.
QEXP : Quaternion number exponentiation.
QIMi : Quaternion number imaginary part i.
QIMj : Quaternion number imaginary part j.

QIMk : Quaternion number imaginary part k.
QINT : Quaternion number from integer.
QINV : Quaternion number inverse.
QNEG : Quaternion number negative.
QNREAD : Quaternion number read.
QNWRITE : Quaternion number write.
QONE : Quaternion number one.
QPROD : Quaternion number product.
QQ : Quaternion number quotient.
QRAND : Quaternion number, random.
QRE : Quaternion number real part.
QRN : Quaternion number from rational number.
QRN4 : Quaternion number from 4-tuple of rational numbers.
QSUM : Quaternion number sum.

queue

EMPTYQUE : Empty Queue.
NEWQUE : New Queue.

quotient

ADPIQ : Arbitrary domain polynomial integer quotient.
ADQR : Arbitrary domain quotient and remainder.
ADQUOT : Arbitrary domain quotient.
AFPAFQ : Algebraic number field polynomial algebraic number field element quotient.
AFPQR : Algebraic number field polynomial quotient and remainder.
AFQ : Algebraic number field quotient.
APQ : Arbitrary precision floating point quotient.
CQ : Complex number quotient.
DIIPIQ : Distributive integral polynomial integer quotient.
DIIPQ : Distributive integral polynomial quotient.
DIIPQR : Distributive integral polynomial quotient and remainder.
DINPQ : Distributive non-commutative Polynomial Quotient.
DIPQR : Distributive polynomial quotient and remainder.
DIRPQ : Distributive rational polynomial quotient.
DIRPQR : Distributive rational polynomial quotient and remainder.
DIRPRQ : Distributive rational polynomial rational number quotient.
DQR : Digit quotient and remainder.
EIVIQ : Exterior integral vector integer quotient.
FFQ : Finite field quotient.
IDQ : Integer-digit quotient.
IDQR : Integer-digit quotient and remainder.
IPIQ : Integral polynomial integer quotient.
IPQ : Integral polynomial quotient.
IPQR : Integral polynomial quotient and remainder.
IQ : Integer quotient.
IQR : Integer quotient and remainder.
IVSQ : Integer vector scalar quotient.
MASQREM : Quotient and remainder.
MDQ : Modular digit quotient.
MIQ : Modular integer quotient.
MIUPQR : Modular integral univariate polynomial quotient and remainder.

MMPIQR : Modular monic polynomial mod ideal quotient and remainder.
MPQ : Modular polynomial quotient.
MPQR : Modular polynomial quotient and remainder.
MPUQ : Modular polynomial univariate quotient.
OQ : Octonion number quotient.
QQ : Quaternion number quotient.
RFQ : Rational function quotient.
RNQ : Rational number quotient.
RNVQ : Rational number vector quotient.
RNVQF : Rational number vector quotient.
RPQR : Rational polynomial quotient and remainder.
RRIPIQ : Real root integral polynomial integral quotient.
RRIPQSUM : Real root integral polynomial quotient sum.
SetQrFunc : Set quotient and remainder function in domain.
SetQuotFunc : Set quotient function in domain.

rabinowitsch

rabinowitsch : rabinowitsch.

radical

RadicalMember : radical membership test.

raise

raise : Raise signal s.

random

BITRAN : Bit, random.
CRAND : Complex number, random.
DIIPRA : Distributive integral polynomial random.
DIIRAS : Distributive integral polynomial random sparse exponent vector.
DIRPRA : Distributive rational polynomial random.
DIRRAS : Distributive rational polynomial, random sparse exponent vector.
DRAN : Digit, random.
DRANN : Digit, random non-negative.
EVRAND : Exponent vector random.
EVRASP : Exponent vector random.
FFRAND : Finite field element, random.
HIPRAN : Homogeneous integral polynomial random.
IPRAN : Integral polynomial random.
IPRAN : Integral polynomial, random.
IRAND : Integer, random.
IVRAND : Integer vector random.
MDRAN : Modular digit, random.
MIPRAN : Modular integral polynomial, random.
MIRAN : Modular integer, random.
MIRAND : Matrix random.
MPRAN : Modular polynomial, random.
MUPRAN : Modular univariate polynomial, random.
ORAND : Octonion number, random.
PARTR : Partition, random.
PERMR : Permutation, random.
QRAND : Quaternion number, random.
RNRAND : Rational number, random.

rang

MRANG : Matrix rang.

rational

ADRFFADIP : arbitrary domain rational function from arbitrary domain intgeral
AFFRN : Algebraic number field element from rational number.
AFPFRP : Algebraic number field polynomial from rational polynomial.
AFUPSR : Algebraic number field univariate polynomial, sign at a rational
APFRN : Arbitrary precision floating point from rational number.
CP2 : UGB linear form product with rational exponent vector list 2.
CQ2 : UGB linear form product with rational exponent vector list.
CRN : Complex number from rational number.
CRNP : Complex number from pair of rational numbers.
DEGRE : UGB total degree of a list of rational exponent vectors.
DFP : UGB distributive rational polynomial difference.
DIFF : UGB difference set for rational exponent vector list.
DIFF1 : UGB difference set for two rational exponent vector list.
DIIFRP : Distributive integral polynomial from rational polynomial.
DIILFR : Distributive integral polynomial list from rational polynomial list.
DIILFRCD : DIP integral list from DIP rational list using common denominator.
DINCCO : Distributive rational non-commutative polynomial commutator.
DINCCP : Distributive rational non-commutative polynomial center polynomial.
DINCCPpre : Distributive rational non-commutative polynomial center polynomial preparation.
DINLMPG : Distributive non-commutative left rational minimal polynomial for a G basis.
DINLMPL : Distributive non-commutative left rational minimal polynomial list for a G basis.
DIPRF : DIP Rational Function Definition Module.
DIPRN : DIP Rational Definition Module.
DIPRNGB : DIP Rational Groebner Bases Definition Module.
DIPRNIB : DIP Rational Numbers Polynomial Definition Module in the sense of Janet.
DIPRNPOL : DIP Rational Number Polynomial Definition Module.
DIREGB : Distributive rational polynomials extended groebner basis.
DIRFAC : Distributive rational polynomial factorisation.
DIRFIP : Distributive rational polynomial from integral polynomial.
DIRGBA : Distributive rational polynomial groebner basis augmentation.
DIRGBR : Distributive rational polynomial groebner basis recursion.
DIRGZS : Distributive rational Groebner base zero set.
DIRLCT : Distributive rational polynomial list ideal containment test.
DIRLIP : Distributive rational polynomial list ideal product.
DIRLIS : Distributive rational polynomial list irreducible set.
DIRLISJ : Distributive rational polynomial list irreducible set.
DIRLPD : DIP rational polynomial ideal primary ideal decomposition.
DIRLPI : Distributive rational polynomial list primary ideal.
DIRLPW : DIP rational polynomial ideal primary ideal decomposition write.
DIRLRD : Distributive rational polynomial list read.
DIRLWR : Distributive rational polynomial list write.
DIRMPG : Distributive rational minimal polynomial for a groebner basis.
DIRPAB : Distributive rational polynomial absolute value.
DIRPCOM : Distributive rational polynom complete system.

DIRPDA : DIP rational polynomial ideal primary ideal decomposition over Q(alpha).
DIRPDF : Distributive rational polynomial difference.
DIRPDM : Distributive rational polynomial derivation main variable.
DIRPDR : Distributive rational polynomial derivation.
DIRPEM : Distributive rational polynomial evaluation of main variable.
DIRPES : Distributive rational polynomial elementary symmetric functions.
DIRPEV : Distributive rational polynomial evaluation of the i-th variable.
DIRPEX : Distributive rational polynomial exponentiation.
DIRPFT : Distributive rational polynomial from term.
DIRPGB : Distributive rational polynomials groebner basis.
DIRPHD : Distributive rational polynomial higher derivation.
DIRPIB2 : Distributive rational polynom involutive basis.
DIRPLS : Distributive rational polynomial list sum.
DIRPMC : Distributive rational polynomial monic.
DIRPMN : Distributive rational polynomial maximum norm.
DIRPNF : Distributive rational polynomial normal form.
DIRPNFJ : Distributive rational polynomial normal form in the sense of Janet.
DIRPNG : Distributive rational polynomial negative.
DIRPON : Distributive rational polynomial one.
DIRPPR : Distributive rational polynomial product.
DIRPQ : Distributive rational polynomial quotient.
DIRPQR : Distributive rational polynomial quotient and remainder.
DIRPRA : Distributive rational polynomial random.
DIRPRD : Distributive rational polynomial read.
DIRPRP : Distributive rational polynomial rational number product.
DIRPRP : Distributive rational polynomial rational number product.
DIRPRQ : Distributive rational polynomial rational number quotient.
DIRPRQ : Distributive rational polynomial rational number quotient.
DIRPSE : Distributive rational polynomial symm.
DIRPSG : Distributive rational polynomial sign.
DIRPSM : Distributive rational polynomial sum.
DIRPSN : Distributive rational polynomial sum norm.
DIRPSO : Distributive rational polynomial sort.
DIRPSP : Distributive rational polynomial S polynomial.
DIRPSR : Distributive rational polynomial symmetric function reduction.
DIRPSU : Distributive rational polynomial substitution.
DIRPSV : Distributive rational polynomial substitution for main variable.
DIRPTM : Distributive rational polynomial translation main variable.
DIRPTR : Distributive rational polynomial translation.
DIRPWR : Distributive rational polynomial write.
DIRPWV : Distributive rational polynomial write with standard variable
DIRRAS : Distributive rational polynomial, random sparse exponent vector.
DomLoadRF : Domain load rational function.
DomLoadRN : Domain load rational number.
DomLoadRP : Domain load rational polynomials .
DOMRF : MAS Domain Rational Function Definition Module.
DOMRN : MAS Domain Rational Number Definition Module.
DOMRP : MAS Domain Rational Polynomial Definition Module.
EVLRNBSO : Rational exponent vector list bubble sort.
EVRNC : Rational exponent vector compare.
EVRNGL : Rational exponent vector inverse graded lexicographical compare.
FFRN : Floating point from rational number.

GRNBAS : G-Symmetric Rational Base Construction.
GRNCHK : G-Symmetric Rational Polynomial Check.
GRNCHKBAS : G-Symmetric Rational Base Check.
GRNCUT : G-Symmetric Rational Polynomial Cut.
GRNGGB : G-Symmetric Rational Base Construction (Buchberger-Algorithm).
GRNOPL : G-Symmetric Rational Orbit Polynomial List.
GRNORP : G-Symmetric Rational Orbit Polynomial.
GRNRED : G-Symmetric Rational Polynomial Reduction.
GSRDREAD : G-symmetric rational descriptor read.
GSRREAD : G-symmetric rational polynomial read.
GSYMFURN : G-Symmetric Rational Polynomial System Definition Module.
IMFRNM : Integer matrix from rational number matrix.
IMFRNM1 : Integer matrix from rational number matrix.
IPFRP : Integral polynomial from rational polynomial.
IPSRP : Integral polynomial similiar to rational polynomial.
IUPBEI : Integral univariate polynomial binary rational evaluation, integer output.
IUPBES : Integral univariate polynomial binary rational evaluation of sign.
IUPBRE : Integral univariate polynomial binary rational evaluation.
IUPMRN : Integral univariate polynomial minimal polynomial of a rational number.
IVFRNV : Integer vector from rational number vector.
IVFRNV1 : Integer vector from rational number vector.
LINALGRN : MAS Linear Algebra Rational Number Definition Module.
LRNBMS : List of rational numbers bubble-merge sort.
LRNBS : List of rational numbers bubble sort.
LRNM : List of rational numbers merge.
LRNWRIT : List of rational numbers write.
MAKERN : UGB rational exponent vector list from integer ev list.
MASRN : MAS Rational Number Definition Module.
MKSET : UGB rational exponent vector list difference list.
NULRNV : Rational number vector null test.
ORN : Octonion number from rational number.
ORNP : Octonion number from pair of rational numbers.
PDIF : UGB rational exponent vector list difference list, incremental.
QRN : Quaternion number from rational number.
QRN4 : Quaternion number from 4-tuple of rational numbers.
RFDDADV : rational function domain descriptor advance.
RFDDFIPDD : rational function domain descriptor from integral polynomial domain
RFDEN : Rational function denominator.
RFDIF : Rational function difference.
RFEXP : Rational function exponentiation.
RFFIP : Rational function from integral polynomial.
RFINV : Rational function inverse.
RFNEG : Rational function negative.
RFNOV : Rational function number of variables.
RFNUM : Rational function numerator.
RFONE : Rational function one.
RFPROD : Rational function product.
RFQ : Rational function quotient.
RFREAD : Rational function read.
RFRED : Rational function reduction to lowest terms.
RFSIGN : Rational function sign.

RFSUM : Rational function sum.
RFWRIT : Rational function write.
RIB : Rational interval bisection.
RILC : Rational interval length comparison.
RINT : Rational interval normalizing transformation.
RIRNP : Rational interval rational number product.
RIRNP : Rational interval rational number product.
RIRWRT : Rational interval refinement write.
RNABS : Rational number absolute value.
RNBCR : Rational number binary common representation.
RNCEIL : Rational number, ceiling of.
RNCOMP : Rational number comparison.
RNDEN : Rational number denominator.
RNDIF : Rational number difference.
RNDRD : Rational number decimal read.
RNDRD : Rational number decimal read.
RNDWR : Rational number decimal write.
RNDWR : Rational number decimal write.
RNDWRS : Rational number decimal write special.
RNEXP : Rational number exponentiation.
RNFAP : Rational number from arbitrary precision floating point.
RNFCL2 : Rational number floor and ceiling of logarithm, base 2.
RNFF : Rational number from floating point.
RNFLOR : Rational number, floor of.
RNINT : Rational number from integer.
RNINV : Rational number inverse.
RNMAX : Rational number maximum.
RNMDET : Rational number matrix determinant, using Gaussian elimination.
RNMDETL : Rational number matrix determinant, using Laplace expansion.
RNMDIF : Rational number matrix difference.
RNMFIM : Rational number matrix from integer matrix.
RNMGE : Rational number matrix Gaussian elimination.
RNMGELUD : Rational number matrix Gaussian elimination LU-decomposition.
RNMHILBERT : Rational number matrix Hilbert.
RNMINV : Rational number matrix inversion.
RNMINVI : Rational number matrix inversion, integer algorithm.
RNMLT : Rational matrix lower triangular matrix transformation.
RNMMAX : Rational number matrix maximum norm.
RNMPROD : Rational number matrix product.
RNMREAD : Rational number matrix read.
RNMSDS : Rational number matrix solve decomposed system.
RNMSUM : Rational number matrix sum.
RNMUNS : Rational number matrix upper triangular matrix solution null space.
RNMUT : Rational matrix upper triangular matrix transformation.
RNMWRITE : Rational number matrix write.
RNNEG : Rational number negative.
RNNUM : Rational number numerator.
RNONE : Rational number one.
RNP2 : Rational number power of 2.
RNPROD : Rational number product.
RNQ : Rational number quotient.
RNRAND : Rational number, random.
RNREAD : Rational number read.

RNRED : Rational number reduction to lowest terms.
RNSIGN : Rational number sign.
RNSMPROD : Rational number scalar and matrix product.
RNSUM : Rational number sum.
RNSVPROD : Rational number vector product with scalar.
RNUM : Rational number unit matrix.
RNVABS : Rational number list absolute values.
RNVDIF : Rational number vector difference.
RNVDIF : UGB rational exponent vector difference.
RNVFIV : Rational number vector from integer vector.
RNVLC : Rational number vector linear combination.
RNVMAX : Rational number vector maximum norm.
RNVQ : Rational number vector quotient.
RNVQF : Rational number vector quotient.
RNVREAD : Rational number vector read.
RNVSPROD : Rational number vector scalar product.
RNVSSUM : Rational number vector scalar sum.
RNVSVPROD : Rational number vector scalar vector product.
RNVSVSUM : Rational number vector scalar sum.
RNVVPROD : Rational number vector vector product.
RNVVSUM : Rational number vector vector sum.
RNVWRITE : Rational number vector write.
RNWRIT : Rational number write.
RPABS : Rational polynomial absolute value.
RPAFME : Rational polynomial, algebraic number field multiple evaluation.
RPBLGS : Rational polynomial base coefficients least common multiple, greatest common divisor, and sign.
RPCONST : Rational polynomial constant.
RPDIF : Rational polynomial difference.
RPDMV : Rational polynomial derivative, main variable.
RPEMV : Rational polynomial evaluation, main variable.
RPEXP : Rational polynomial exponentiation.
RPFIP : Rational polynomial from integral polynomial.
RPIMV : Rational polynomial integration, main variable.
RPLWRS : Rational polynomial list write.
RPMAIP : Rational polynomial monic associate of integral polynomial.
RPMON : Rational polynomial monic.
RPNEG : Rational polynomial negative.
RPONE : Rational polynomial one.
RPPROD : Rational polynomial product.
RPQR : Rational polynomial quotient and remainder.
RPREAD : Rational polynomial read.
RPRNP : Rational polynomial rational number product.
RPRNP : Rational polynomial rational number product.
RPSIGN : Rational polynomial sign.
RPSUM : Rational polynomial sum.
RPWRIT : Rational polynomial write.
RPWRTS : Rational polynomial write.
RUPEGC : Rational univariate polynomial extended greatest common divisor.
RUPGCD : Rational univariate polynomial greatest common divisor.
RUPHEG : Rational univariate polynomial half-extended greatest common divisor.
RUPLCM : Rational univariate polynomial least common multiple.

RUPMRN : Rational univariate polynomial minimal polynomial of a rational number.
RUPMRN : Rational univariate polynomial minimal polynomial of a rational number.
SACRN : SAC Rational Number Definition Module.
SACRPOL : SAC Rational Polynomial Definition Module.
SCMULT : UGB rational exponent vector rational number product.
SCMULT : UGB rational exponent vector rational number product.
SCPROD : UGB rational exponent vector scalar product.
SFP : UGB distributive rational polynomial sum.
SKPRO2 : UGB rational exponent vector scalar product with integer ev.
TFDIRP : Term from distributive rational polynomial.
TIPRNGB : DIP Rational Extended Groebner Bases Definition Module.

rational-weighted

DIPRWTDG : Distributive polynomial rational-weighted total degree.
EVRWTDEG : Exponent vector rational-weighted total degree.

re-sort

DIPLRS : Distributive polynomial list re-sort.

read

ADDDREAD : Arbitrary domain, domain descriptor read.
ADREAD : Arbitrary domain read.
AREAD : Atom read.
CdpRead : Case distinction and polynomial set read.
CdRead : Case distinction read.
CDREAD : Complex number decimal read.
CNREAD : Complex number read.
CondPRead : Condition part read.
CONDRD : Conditions read.
CondRead : Condition read.
CREAD : Character read.
CREADB : Character read, skipping blanks.
DIILRD : Distributive integral polynomial list read.
DIIPRD : Distributive integral polynomial read.
DILRD : Distributive polynomial list read.
DINLRD : Distributive non-commutative polynomial list read.
DINPRD : Distributive non-commutative polynomial read.
DIREAD : Distributive polynomial read.
DIRLRD : Distributive rational polynomial list read.
DIRPRD : Distributive rational polynomial read.
DVREAD : Polynom descriptor read.
EPREAD : Exponent read.
EXECRD : Exec read.
EXECRD : UGB execution options read.
FFREAD : Finite field read.
FORIREAD : formula infix read.
FORPREAD : formula prefix read.
FORRDLVAR : formula read list of variables.
FORRDVAR : formula read variable.

GREAD : Gamma-integer read.
GSDREAD : G-symmetric descriptor read.
GSPREAD : G-symmetric polynomial read.
GSRDREAD : G-symmetric rational descriptor read.
GSRREAD : G-symmetric rational polynomial read.
GSYPGR : G-Symmetric Permutation Group Read.
IPREAD : Integral polynomial read.
IREAD : Integer read.
KEYREAD : key read.
LREAD : List read.
masReadL : MAS Read Line.
masReadOpen : MAS Read Open
MLDIREAD : maslog demonstration infix read.
MLDPREAD : maslog demonstration prefix read.
MREAD : Matrix Read.
NMREAD : Non-Commutative Matrix Read.
ODREAD : Octonion number decimal read.
ONREAD : Octonion number read.
OPREAD : UGB options and parameter read.
OREAD : Object read.
PQIREAD : polynomial equation infix read.
PQPREAD : polynomial equation read.
pqreadaf : polynomial equation read atomic formula.
PREAD : UGB polynomial read.
QDREAD : Quaternion number decimal read.
QNREAD : Quaternion number read.
RDPAR : UGB read parameter.
RDSYS : Read polynomial systems.
RFREAD : Rational function read.
RNDRD : Rational number decimal read.
RNDRD : Rational number decimal read.
RNMREAD : Rational number matrix read.
RNREAD : Rational number read.
RNVREAD : Rational number vector read.
RPREAD : Rational polynomial read.
SetDdrdFunc : Set domain descriptor read function in domain.
SetReadFunc : Set read function in domain.
SREAD : Symbol read.
SREAD : Symbol read.
SREAD1 : Symbol read, 1.
SREAD1 : Symbol read, 1.
SUBSGR : Substitution Group Read.
TFIREAD : type formula infix read.
UREAD : Universal read.
UREAD : Universal read.
UREAD : Universal read.
VdRead : Variable list and domain descriptor read.
VLREAD : Variable list read.
VREAD : Variable read.

readaption

ALFRA : Automatic Linear Form Readaption.
TR : T Readaption.

readline

masreadline : readline Foreign Module.

real

AFPBRI : Algebraic number field polynomial basis real root isolation.
AFPCLL : Algebraic number field polynomial real root isolation, Collins-Loos
AFPMPR : Algebraic number field polynomial minimal polynomial of a real root.
AFPRCL : Algebraic number field polynomial real root isolation, Collins-Loos algorithm.
AFPRII : Algebraic number field polynomial real root isolation induction.
AFPRLS : Algebraic number field polynomial real root list separation.
AFPRRI : Algebraic number field polynomial relative real root isolation.
AFPRRS : Algebraic number field polynomial real root separation.
CRE : Complex number real part.
DIPROOT : DIP Ideal Real Root System Definition Module.
DIROWR : Distributive polynomial system real root write.
GBZSET : Groebner base real zero set of zero dimensional ideal.
InitExternalsM : Initialize external compiled real root procedures.
IPLRRI : Integral polynomial list real root isolation.
IPRCH : Integral polynomial real root calculation, high precision.
IPRCHS : Integral polynomial real root calculation, high-precision special.
IPRCN1 : Integral polynomial real root calculation, 1 root.
IPRCNP : Integral polynomial real root calculation, newton method preparation.
IPRICL : Integral polynomial real root isolation, Collins-Loos algorithm.
IPRIM : Integral polynomial real root isolation, modified Uspensky method.
IPRIMO : Integral polynomial real root isolation, modified Uspensky method, open interval.
IPRIMS : Integral polynomial real root isolation, modified Uspensky method, standard interval.
IPRIMU : Integral polynomial real root isolation, modified Uspensky method, unit interval.
IPRIU : Integral polynomial real root isolation, Uspensky method.
IPRIUP : Integral polynomial real root isolation, Uspensky method, positive roots.
IPRRII : Integral polynomial real root isolation induction.
IPRRLS : Integral polynomial real root list separation.
IPRRRI : Integral polynomial relative real root isolation.
IPRRS : Integral polynomial real root separation.
IPSRM : Integral polynomial strong real root isolation, modified Uspensky method.
IPSRMS : Integral polynomial strong real root isolation, modified Uspensky method, standard interval.
ORE : Octonion number real part.
QRE : Quaternion number real part.
RQEOPTSET : real quantifier elimination options set.
RQEOPTWR : real quantifier elimination option write.
RQEPRRC : Real Quantifier Elimination with Parametric Real Root Count.
RQEPRRC : Real Quantifier Elimination with Parametric Real Root Count.
RQEQE : real quantifier elimination quantifier elimination.
RRADCOUNT : Real root arbirary domain count.
RRADNFORM : Real root arbitrary domain normal form.
RRADOM : Real Root Arbitrary Domain Definition Module.

RRADPOLMATRIX : Real root arbitrary domain polynomial matrix.
RRADQUADFORM : Real root arbitrary domain quadratic form.
RRADSTRCONST : Real root arbitrary domain structure constants.
RRADVARMATRICES : Real root arbitrary domain multiplication matrices of variables.
RRCSR : Real root count solve and reduce.
RRICOUNT : Real root integral count.
RRINFORM : Real root integral normal form.
RRINT : Real Root Integral Definition Module.
RRIPIQ : Real root integral polynomial integral quotient.
RRIPOLMATRIX : Real root integral polynomial matrix.
RRIPQSUM : Real root integral polynomial quotient sum.
RRIQUADFORM : Real root integral quadratic form.
RRISTRCONST : Real root integral structure constants.
RRIVARMATRICES : Real root integral multiplication matrices of variables.
RRMMULT : Real root matrix of multiplication.
RRREDTERMS : Real root reduced terms.
RRREDTEST : Real root reducibility test.
RRUADCOUNT : Real root univariate arbirary domain count.
RRUADOM : Real Root Univariate Arbitrary Domain Definition Module.
RRUADPOLTOVEC : Real root univariate arbitrary domain polynomial to vector.
RRUADQUADFORM : Real root univariate arbitrary domain quadratic form.
RRUADSTRCONST : Real root univariate arbitrary domain structure constants.
RRUICOUNT : Real root univariate integral count.
RRUINT : Real Root Univariate Integral Definition Module.
RRUIPOLTOVEC : Real root univariate integral polynomial to vector.
RRUIQUADFORM : Real root univariate integral quadratic form.
RRUISTRCONST : Real root univariate integral structure constants.
RRVTEXT : Real root vector of tupels extension.
RRZDIM : Real root zero-dimensional test.
SACROOT : SAC Polynomial Real Root Definition Module.

reciprocal

PRT : Polynomial reciprocal transformation.

recursion

DIRGBR : Distributive rational polynomial groebner basis recursion.
TfRecBasis : type formula recursion basis.

recursive

DILFPFL : Groebner bases and related procedures for recursive integral polynomials.
DIPLFPFL : Distributive polynomial list from recursive polynomial
PFLDIPL : Recursive polynomial list (with domain-descriptor) from distributive
XDIPFPF : Distributive polynomial from recursive polynomial (with domain-descriptor).
XPFDIP : Recursive polynomial (with domain-descriptor) from distributive polynomial.

red

ColRed : Colouring red.
REDSRT : Red terms sort.

reduce

CGBGLOBRED : Comprehensive Groebner basis global reduce.
DIDPREDDGB : Distributive domain polynomial reduce D-groebner base.
DIIPREDDGB : Distributive integral polynomial reduce D-groebner base.
DIPRLF : distributive polynomials reduce list of polynomials with factor.
GSYSRED : Reduce Groebner system.
RRCSR : Real root count solve and reduce.

reduce-Exponent

SetReduceExp : Set Reduce-Exponent for procedure GroebnerBases2:

reduced

IPRPRS : Integral polynomial reduced polynomial remainder sequence.
MKSP1 : UGB compute next non-zero reduced S-polynomial.
RRREDTERMS : Real root reduced terms.
WRRCGB : Write reduced comprehensive-groebner-basis.

reducibility

RRREDTEST : Real root reducibility test.

reducible

ADIredG : Arbitrary domain polynomial set I reducible modulo G.

reduct

REDUCT : Reduct.

reductas

DIGISR : DIP G base index search for extension reductas.

reduction

DGBRED : Discrete Groebner Base Reduction.
DIPADGBRED : distributive polynomial groebner basis reduction.
DIRPSR : Distributive rational polynomial symmetric function reduction.
EVR : Exponent Vector Reduction.
GBTMRED : GBTM Reduction.
GINRED : G-Symmetric Integral Polynomial Reduction.
GLOBRE : Global reduction.
GRED : Parametric reduction.
GRNRED : G-Symmetric Rational Polynomial Reduction.
GSRED : Groebner-System reduction.
NLDGBRED : Non-Commutative Discrete Groebner Base Reduction.
NLRCSP : Non-Commutative Reduction Chain of S-Polynomials.
NLRCSPR : Reduction Chain of S-Polynomials with Remainder.
RCSP : Reduction Chain of S-Polynomials.
RCSPR : Reduction Chain of S-Polynomials with Remainder.
RDNORM : Reduction normalform.
REFIND : Reduction find polynomial.
RFRED : Rational function reduction to lowest terms.
RNRED : Rational number reduction to lowest terms.
VIERED : Vector of integers, element reduction.

reductum

DIPMRD : Distributive polynomial monomial reductum.
DIPRED : Distributive polynomial reductum.
PRED : Polynomial reductum.
RED : Reductum.
RED2 : Reductum 2.
RED3 : Reductum 3.
RED4 : Reductum 4.
REDUCT : Reductum.
SRED : Set reductum.

reference

GENARRAY : Generate array reference symbol.

refinement

DINTSR : DIP normalized tupel separation refinement.
ISFPIR : Integral squarefree polynomial isolating interval refinement.
RIRWRT : Rational interval refinement write.

register

getstck : Get contents of stack register.
gettoc : Get contents of toc register.

related

DILFPFL : Groebner bases and related procedures for recursive integral polynomials.

relation

NextRel : next relation.
pqgrel : polynomial equation get relation symbol.
tfgrel : type formula get relation symbol.

relations

PQCRELAND : contract relations or.
PQCRELOR : contract relations or.

relative

AFPRRI : Algebraic number field polynomial relative real root isolation.
DIPCNSTR : distributive polynomial constant relative to variables.
IPRRRI : Integral polynomial relative real root isolation.

relatively

EVCNSTR : exponent vector constant relatively.

remainder

ADQR : Arbitrary domain quotient and remainder.
ADREM : Arbitrary domain remainder.
AFPQR : Algebraic number field polynomial quotient and remainder.
DIIPQR : Distributive integral polynomial quotient and remainder.
DIPQR : Distributive polynomial quotient and remainder.
DIRPQR : Distributive rational polynomial quotient and remainder.
DMUPNR : Dense modular univariate polynomial natural remainder.
DQR : Digit quotient and remainder.
IDQR : Integer-digit quotient and remainder.
IDREM : Integer-digit remainder.
IPCRA : Integral polynomial chinese remainder algorithm.
IPQR : Integral polynomial quotient and remainder.
IPRPRS : Integral polynomial reduced polynomial remainder sequence.
IPSPRS : Integral polynomial subresultant polynomial remainder sequence.
IQR : Integer quotient and remainder.
IREM : Integer remainder.
MASQREM : Quotient and remainder.
MASREM : Remainder.
MDCRA : Modular digit chinese remainder algorithm.
MDLCRA : Modular digit list chinese remainder algorithm.
MIDCRA : Modular integer digit chinese remainder algorithm.
MIUPQR : Modular integral univariate polynomial quotient and remainder.
MMPIQR : Modular monic polynomial mod ideal quotient and remainder.
MPQR : Modular polynomial quotient and remainder.
MPSPRS : Modular polynomial subresultant polynomial remainder sequence.
NLRCSPR : Reduction Chain of S-Polynomials with Remainder.
RCSPR : Reduction Chain of S-Polynomials with Remainder.
RPQR : Rational polynomial quotient and remainder.
SetQrFunc : Set quotient and remainder function in domain.
SetRemFunc : Set remainder function in domain.

remove

ADRMDD : arbitrary domain remove domain descriptor informations.
GLEXTP : Global extraneous polynomials remove.
NOEPRM : Noether Remove.
REMPRP : Remove property.
REMPRP : Remove property.
REXTP : Remove extraneous polynomials.
RMGRT : Remove green terms.

replace

FORREPAFS : formula replace atomic formulas.

represent

ANREPE : Algebraic number represent element of a primitive extension.

representation

CopyRep : Copy representation.
FullRep : Full representation.
GetRep : Get representation.
MASREP : MAS Representation Definition Module.
NewRep : New representation.
RNBCR : Rational number binary common representation.
SetRep : Set representation.
StepRep : Step through representation.

representatives

SDR : System of distinct representatives.

required

UPDATE : Update of extended ideal basis and extended critical pair list as required

RES

MASPGCD : MAS Polynomial GCD and RES System Definition Module.
SACPGCD : SAC Polynomial GCD and RES System Definition Module.

reset

EvordReset : Reset evord.
ValisReset : Reset valis.

residue

FRESL : Fermat residue list.
FRLSM : Fermat residue list, single modulus.

respect

MCPMV : Matrix of coefficients of polynomials, with respect to main variable.

restore

FORVTRESTORE : formula variable table restore.

resultant

IPRES : Integral polynomial resultant.
IUPRC : Integral univariate polynomial resultant and cofactor.
MPRES : Modular polynomial resultant.
MUPRC : Modular univariate polynomial resultant and cofactor.
MUPRES : Modular univariate polynomial resultant.
UIPRES : Univariate integral polynomials resultant.
UIPRS1 : Univariate integral polynomials resultant 1.

returns

ClocK : ClocK returns milliseconds of the processes cpu-time.

right

DNNRGB : distributive polynomials non-noetherian right Groebner base.
DNRCPL : distributive polynomial non-noetherian right construct pair list.
DNRUPL : distributive polynomial non-noetherian right update pair list.
EIVIRP : Exterior integral vector inner right product.
ILIRPR : Index list inner right product.
IMRTPROD : Integral matrix right tensor product.

ring

DILPFDIL : distributive polynomials over polynomial ring list from distributive
DIPFDIPP : distributive polynomial from distributive polynomial over polynomial ring.
DIPPFDIP : distributive polynomial over polynomial ring from distributive polynomial.
MPPFMP : monomial of polynomial ring over polynomial ring from monomial of
MPPFMP : monomial of polynomial ring over polynomial ring from monomial of
PQPRING : polynomial equation polynomial ring.
PQPRINGWR : polynomial equation polynomial ring write.
RQEPRRC : This global variable holds information over the actual polynomial ring

rings

DINNGB : DIP Groebner bases for non noetherian polynomial rings.

root

AFPBRI : Algebraic number field polynomial basis real root isolation.
AFPCLL : Algebraic number field polynomial real root isolation, Collins-Loos
AFPMPR : Algebraic number field polynomial minimal polynomial of a real root.
AFPRCL : Algebraic number field polynomial real root isolation, Collins-Loos algorithm.
AFPRII : Algebraic number field polynomial real root isolation induction.
AFPRLS : Algebraic number field polynomial real root list separation.
AFPRRI : Algebraic number field polynomial relative real root isolation.
AFPRRS : Algebraic number field polynomial real root separation.
AFUPRB : Algebraic number field univariate polynomial root bound.
AFUPRL : Algebraic number field polynomial, root of a linear polynomial.
APROOT : Arbitrary precision floating point n-th root.
DIPROOT : DIP Ideal Real Root System Definition Module.
DIROWR : Distributive polynomial system real root write.
DSQRTF : Digit square root function.
InitExternalsD : Initialize external compiled ideal decomposition and root procedures.
InitExternalsM : Initialize external compiled real root procedures.
IPLRRI : Integral polynomial list real root isolation.
IPRCH : Integral polynomial real root calculation, high precision.
IPRCHS : Integral polynomial real root calculation, high-precision special.
IPRCN1 : Integral polynomial real root calculation, 1 root.
IPRCN1 : Integral polynomial real root calculation, 1 root.
IPRCNP : Integral polynomial real root calculation, newton method preparation.
IPRICL : Integral polynomial real root isolation, Collins-Loos algorithm.
IPRIM : Integral polynomial real root isolation, modified Uspensky method.

IPRIMO : Integral polynomial real root isolation, modified Uspensky method, open interval.
IPRIMS : Integral polynomial real root isolation, modified Uspensky method, standard interval.
IPRIMU : Integral polynomial real root isolation, modified Uspensky method, unit interval.
IPRIU : Integral polynomial real root isolation, Uspensky method.
IPRIUP : Integral polynomial real root isolation, Uspensky method, positive roots.
IPRRII : Integral polynomial real root isolation induction.
IPRRLS : Integral polynomial real root list separation.
IPRRRI : Integral polynomial relative real root isolation.
IPRRS : Integral polynomial real root separation.
IPSRM : Integral polynomial strong real root isolation, modified Uspensky method.
IPSRMS : Integral polynomial strong real root isolation, modified Uspensky method, standard interval.
IROOT : Integer root.
ISQRT : Integer square root.
IUPRB : Integral univariate polynomial root bound.
IUPRLP : Integral univariate polynomial, root of a linear polynomial.
RQEPRRC : Real Quantifier Elimination with Parametric Real Root Count.
RRADCOUNT : Real root arbirary domain count.
RRADNFORM : Real root arbitrary domain normal form.
RRADOM : Real Root Arbitrary Domain Definition Module.
RRADPOLMATRIX : Real root arbitrary domain polynomial matrix.
RRADQUADFORM : Real root arbitrary domain quadratic form.
RRADSTRCONST : Real root arbitrary domain structure constants.
RRADVARMATRICES : Real root arbitrary domain multiplication matrices of variables.
RRCSR : Real root count solve and reduce.
RRICOUNT : Real root integral count.
RRINFORM : Real root integral normal form.
RRINT : Real Root Integral Definition Module.
RRIPIQ : Real root integral polynomial integral quotient.
RRIPOLMATRIX : Real root integral polynomial matrix.
RRIPQSUM : Real root integral polynomial quotient sum.
RRIQUADFORM : Real root integral quadratic form.
RRISTRCONST : Real root integral structure constants.
RRIVARMATRICES : Real root integral multiplication matrices of variables.
RRMMULT : Real root matrix of multiplication.
RRREDTERMS : Real root reduced terms.
RRREDTEST : Real root reducibility test.
RRUADCOUNT : Real root univariate arbirary domain count.
RRUADOM : Real Root Univariate Arbitrary Domain Definition Module.
RRUADPOLTOVEC : Real root univariate arbitrary domain polynomial to vector.
RRUADQUADFORM : Real root univariate arbitrary domain quadratic form.
RRUADSTRCONST : Real root univariate arbitrary domain structure constants.
RRUICOUNT : Real root univariate integral count.
RRUINT : Real Root Univariate Integral Definition Module.
RRUIPOLTOVEC : Real root univariate integral polynomial to vector.
RRUIQUADFORM : Real root univariate integral quadratic form.
RRUISTRCONST : Real root univariate integral structure constants.
RRVTEXT : Real root vector of tupels extension.
RRZDIM : Real root zero-dimensional test.
SACROOT : SAC Polynomial Real Root Definition Module.

roots

IPRIUP : Integral polynomial real root isolation, Uspensky method, positive roots.
IPSR : Integral polynomial specified roots.

rosser

LDSSBR : Linear diophantine system solution, based on Rosser ideas.

rotation

LEROT : List element rotation.

rule

DEFRULE : Define generic rule function.

S-Expresion

MASLISPU : Types, S-Expresion Types and Indicators.

S-expression

SEXPRP : Test if X is a S-expression function.
TYPEOF : Type of S-expression.

S-polynomial

ADPSP : Arbitrary domain polynomial S-polynomial.
ADPSUGSP : Arbitrary domain polynomial normal with sugar strategy S-polynomial.
DIIFSP : Distributive integral function polynom S-polynomial.
DINLSP : Distributive non-commutative polynomial left S-polynomial.
DIPS : distributive polynomial S-polynomial.
DIPSP : Distributive polynomial S-polynomial.
DIRPSP : UGB distributive polynomial S-polynomial.
EDIPSP : Extended distributive polynomial S-polynomial.
EDIPSUGSP : Extended distributive polynomial normal with sugar strategy S-polynomial.
MKSP1 : UGB compute next non-zero reduced S-polynomial.

S-Polynomial

NLSPC : Non-Commutative S-Polynomial with Coefficients.

S-polynomial

SetEDIPSPolynomial : Set the extended distributive S-polynomial function.
SetPSpolFunc : Set polynomial S-polynomial function in domain.
SetPSugSpolFunc : Set polynomial normal with sugar strategy S-polynomial function in domain.

S-Polynomial

SPC : S-Polynomial with Coefficients.

S-Polynomials

NLRCSP : Non-Commutative Reduction Chain of S-Polynomials.
NLRCSPR : Reduction Chain of S-Polynomials with Remainder.
NLSPCEGB : Non-Commutative S-Polynomials with Coefficients and Exponenetvector-Check
NLSPCGB : Non-Commutative S-Polynomials with Coefficients for Groebner Base.
RCSP : Reduction Chain of S-Polynomials.
RCSPR : Reduction Chain of S-Polynomials with Remainder.
SPCEGB : S-Polynomials with Coefficients and Exponenetvector-Check for Groebner Base.
SPCGB : S-Polynomials with Coefficients for Groebner Base.

SAC

SACANF : SAC Algebraic Number Field Definition Module.
SACBIOS : SAC Basic I/O System Definition Module.
SACCOMB : SAC Combinatorical System Definition Module.
SACD : SAC Digit Definition Module.
SACDPOL : SAC Dense Polynomial Definition Module.
SACEXT1 : SAC Extensions 1 Definition Module.
SACEXT2 : SAC Extensions 2 Definition Module.
SACEXT3 : SAC Extensions 3 Definition Module.
SACEXT4 : SAC Extensions 4 Definition Module.
SACEXT5 : SAC Extensions 5 Definition Module.
SACEXT6 : SAC Extensions 6 Definition Module.
SACEXT7 : SAC Extensions 7 Definition Module.
SACEXT8 : SAC Extensions 8 Definition Module.
SACI : SAC Integer Definition Module.
SACIPOL : SAC Integer Polynomial System Definition Module.
SACLDIO : SAC Linear Diophantine Equation System Definition Module.
SACLIST : SAC List Processing Definition Module.
SACM : SAC Modular Digit and Integer Definition Module.
SACMPOL : SAC Modular Polynomial Definition Module.
SACMUFAC : SAC Modular Univariate Polynomial Factorization Definition Module.
SACPFAC : SAC Polynomial Factorization Definition Module.
SACPGCD : SAC Polynomial GCD and RES System Definition Module.
SACPOL : SAC Polynomial System Definition Module.
SACPRIM : SAC Factorization and Prime Number Definition Module.
SACRN : SAC Rational Number Definition Module.
SACROOT : SAC Polynomial Real Root Definition Module.
SACRPOL : SAC Rational Polynomial Definition Module.
SACSET : SAC Set Definition Module.
SACSYM : SAC Symbol System Definition Module.
SACSYM2 : SAC Symbol 2 Definition Module.
SACUPFAC : SAC Univariate Polynomial Factorization Definition Module.

scalar

ADSMPROD : Arbitrary domain scalar and matrix product.
ADVSPROD : Arbitrary domain vector scalar product.
ADVSVPROD : Arbitrary domain vector scalar vector product.
ISMPROD : Integer scalar and matrix product.
IVSPROD : Integer vector scalar product.
IVSQ : Integer vector scalar quotient.
IVSSUM : Integer vector scalar sum.
IVSVPROD : Integer vector scalar and vector product.
IVSVSUM : Integer vector scalar and vector sum.
RNSMPROD : Rational number scalar and matrix product.
RNSVPROD : Rational number vector product with scalar.
RNVSPROD : Rational number vector scalar product.
RNVSSUM : Rational number vector scalar sum.
RNVSVPROD : Rational number vector scalar vector product.
RNVSVSUM : Rational number vector scalar sum.
SCPROD : UGB rational exponent vector scalar product.
SKPRO2 : UGB rational exponent vector scalar product with integer ev.
VISPR : Vector of integers scalar product.

search

DIGISM : DIP G base index search for extension multiple univariats.
DIGISR : DIP G base index search for extension reductas.
ILPDS : Integer large prime divisor search.
IMPDS : Integer medium prime divisor search.
LSRCH : List search.
LSRCHQ : List search equal.
SCOV : Search condition.
STSRCH : Symbol tree search.
STSRCH : Symbol tree search.
VLSRCH : Variable list search.

second

DIRPIB : Second Algorithm for computing the involutive Base for a given F.
ECPPOLY2 : Extended critical pair select the second extended distributive
IPFSD : Integral polynomial factorization, second derivative.
IPSFSD : Integral squarefree factorization, second derivative.
SECOND : Second.

see

INTDIM : See intdim of dip.

segmentation

ISEG : Integer segmentation.

select

DIPSSM : Distributive polynomial sort and select minimal.
ECPLCMHT : Extended critical pair select the least common multiple of head terms.
ECPPOLY1 : Extended critical pair select the first extended distributive polynomial.
ECPPOLY2 : Extended critical pair select the second extended distributive
ECPSELECT : Select an extended critical pair from the extended critical pair list.
ECPSUGAR : Extended critical pair select sugar.
SetDIPIBSelect : Set distributive polynomial involutive base select.
SetDIPIIBSelect : Set Distributive integral polynomial Select.

selection

SetDIPAGBStrategy : Set the DIPAGB strategy option for the extended critical pair selection.
SetECPSelect : Set the extended critical pair selection procedure.
WriteDIPAGBStrategy : Write the DIPAGB strategy option for the extended critical pair selection

selfridge

ISPT : Integer selfridge primality test.

sense

ADEPNFJ : Arbitrary domain extended polynomial normalform in the sense of Janet.
ADPNFJ : Arbitrary domain polynomial normalform in the sense of Janet.
ADPNFJS : Arbitrary domain polynomial normalform in the sense of Janet modulo a set
DIIPNFJ : Distributive integral polynomial normal form in the sense of Janet.
DILISJ : Distributive polynomial list irreducible set in the sense of Janet.
DILNFJ : Distributive Polynomial List normalform in the sense of Janet.
DIPCJ : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPEINFJ : Integral distributive extended polynomial normal form in the sense of Janet.
DIPENFJ : Distributive extended polynomial normal form in the sense of Janet.
DIPIB : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPIIB : DIP Integral Polynomial System Definition Module in the sense of Janet.
DIPINFJ : Integral Distributive polynomial normal form in the sense of Janet.
DIPINFJS : Integral Distributive polynomial normal form in the sense of Janet modulo a
DIPIRLJ : distributive polynomial interreduced list in the sense of Janet.
DIPIRLJ2 : Distributive polynomial list interreduced list in the sense of Janet.
DIPNFJ : Distributive polynomial normal form in the sense of Janet.
DIPNFJS : Distributive polynomial normal form in the sense of Janet modulo a set of
DIPRNIB : DIP Rational Numbers Polynomial Definition Module in the sense of Janet.
DIRPNFJ : Distributive rational polynomial normal form in the sense of Janet.
EVMTJ : Exponent vector multiple test in the sense of Janet.
SetDIPIBISJ : Set distributive involutive base irreducible set in the sense of Janet.

separation

AFPRLS : Algebraic number field polynomial real root list separation.
AFPRRS : Algebraic number field polynomial real root separation.
DINTSR : DIP normalized tupel separation refinement.
DINTSS : DIP normalized tupel strong separation.
IPRRLS : Integral polynomial real root list separation.
IPRRS : Integral polynomial real root separation.

sequence

EVLGIL : Exponent vector list generate for inverse lexicographical sequence.
IPRPRS : Integral polynomial reduced polynomial remainder sequence.
IPSPRS : Integral polynomial subresultant polynomial remainder sequence.
MPSPRS : Modular polynomial subresultant polynomial remainder sequence.

set

ADIredG : Arbitrary domain polynomial set I reducible modulo G.
ADPNFJS : Arbitrary domain polynomial normalform in the sense of Janet modulo a set
Aparse : Parse a set of ALDES-2 declarations and algorithms.
APSPRE : Arbitrary precision floating point set precision.
CdpCd : Case distinction and polynomial set case distinction part.
CdpCons : Case distinction and polynomial set construct.
CdpParts : Case distinction and polynomial set parts.
CdpPs : Case distinction and polynomial set polynomial set part.
CdpPs : Case distinction and polynomial set polynomial set part.
CdpRead : Case distinction and polynomial set read.
CdpVd : Case distinction and polynomial set variable list and domain descriptor
CdpWrite : Case distinction and polynomial set write.
CLF2 : UGB compute linear form from difference set 2.
CLF3 : UGB compute linear form from difference set 3.
COMPLF : UGB compute linear form from difference set.
CSFPAR : Characteristic set from partition.
CSINT : Characteristic set intersection.
CSSUB : Characteristic set subset.
CSUN : Characteristic set union.
CUT : UGB set of cuts.
DIDIMS : Distributive polynomial dimension maximal independent set.
DIFF : UGB difference set for rational exponent vector list.
DIFF1 : UGB difference set for two rational exponent vector list.
DIIFLS : Distributive integral function polynomial list irreducible set.
DIILIS : Distributive integral polynomial list irreducible set.
DIILIS : Distributive integral polynomial list irreducible set.
DIILISJ : Distributive integral polynomial list irreducible set.
DILIS : Distributive polynomial list irreducible set.
DILISJ : Distributive polynomial list irreducible set in the sense of Janet.
DILISJ2 : Distributive polynomial list irreducible set.
DIMIS : Dimension and maximal independent set.
DINLIS : Distributive non-commutative polynomial list left irreducible set.
DIPADIRSET : distributive polynomial arbitrary domain irreducible set.
DIPMST : Distributive polynomial monomial set.

DIPNFJS : Distributive polynomial normal form in the sense of Janet modulo a set of
DIRGZS : Distributive rational Groebner base zero set.
DIRLIS : Distributive rational polynomial list irreducible set.
DIRLISJ : Distributive rational polynomial list irreducible set.
DITFZS : DIP tupel from zero set.
DITSPL : DIP zero set tupel split.
EvordSet : EVORD set.
EVSSPROD : Exponent vektor set sorted product.
GBZSET : Groebner base real zero set of zero dimensional ideal.
GENINDEX : Generate index set.
IUPFDS : Integral univariate polynomial factor degree set.
MASSET : MAS Set Definition Module.
MSET : Matrix set.
MVSET : modula variable set.
NSET : Normalform set.
PFILDS : Integral polynomial list d-irreducible set.
PFILS : Integral polynomial list irreducible set.
RQEOPTSET : real quantifier elimination options set.
SACSET : SAC Set Definition Module.
SCOMP : Set composition.
SDIFF : Set difference.
SetAbsFunc : Set absolute value function in domain.
SETADD : set add element.
SetAdd : Set add.
SetAddQ : Set add equal.
SetBranchProc : Set Branch-Procedure for procedure GroebnerBases2:
SetCnstFunc : Set constant test function in domain.
SETCOL : Set colour.
SetCompFunc : Set comparison function in domain.
SetComplement : Set complement.
SetComplementQ : set complement.
SetConvFunc : Set conversion function in domain.
SetCPExtend : Set the critical pair extension function.
SetDCGBopt : Set options for decompositional groebner bases.
SetDCIBDecomp : Set decompositional involutive base decomposition.
SetDCIBdepth : Set decompositional involutive base depth of tree.
SetDCIBopt : Set decompositional involutive base options.
SetDCIBTraceLevel : Set Decompositional involutive base Trace Level.
SetDCIBVarOrdOpt : Set decompositional involutive base variable order option.
SetDdrdFunc : Set domain descriptor read function in domain.
SetDdwrFunc : Set domain descriptor write function in domain.
SetDecompProc : Set Decomposition-Procedure for decompositional groebner bases:
SetDifFunc : Set difference function in domain.
SetDIPAGBOptions : Set the trace flag, the strategy and the variable weight list of the
SetDIPAGBStrategy : Set the DIPAGB strategy option for the extended critical pair selection.
SetDIPAGBTraceFlag : Set the DIPAGB trace flag.
SetDIPAGBVariableWeight : Set the DIPAGB variable weight list for the normal with sugar strategy.
SetDIPExtend : Set the distributive polynomial extension function.
SetDIPIBCancel : Set distributive polynomial involutive base cancel.
SetDIPIBCrit : Set distributive polynomial involutive base critera.

SetDIPIBISJ : Set distributive involutive base irreducible set in the sense of Janet.
SetDIPIBISJ : Set distributive involutive base irreducible set in the sense of Janet.
SetDIPIBopt : Set distributive polynomial involutive base options.
SetDIPIBSelect : Set distributive polynomial involutive base select.
SetDIPIBTraceLevel : Set Trace Level.
SetDIPIIBSelect : Set Distributive integral polynomial Select.
SetECPInsert : Set the extended critical pair insertion function.
SetECPSelect : Set the extended critical pair selection procedure.
SetECPWrite : Set the extended critical pair write procedure.
SetEDIPNormalform : Set the extended distributive polynomial normalform function.
SetEDIPSPolynomial : Set the extended distributive S-polynomial function.
SetEDIPUnExtend : Set the extended distributive polynomial un-extension function.
SetEDIPWrite : Set the extended distributive polynomial write procedure.
SetElementP : Set element predicate.
SetElementPQ : Set element predicate equal.
SetExpFunc : Set exponential function in domain.
SetFacSugar : Set Factor-Sugar for procedure GroebnerBases1:
SetFactFunc : Set factorization function in domain.
SetFactoFunc : Set factorization with variable order optimization function in domain.
SetFIntFunc : Set from integer function in domain.
SetFIPolFunc : Set from integral polynomial function in domain.
SetGcdcFunc : Set gcd-and-cofactors function in domain.
SetGcdeFunc : Set gcd-and-lin-combination function in domain.
SetGcdFunc : Set gcd function in domain.
SetInit : Set the initialization procedure.
SetInsert : Set insert.
SetInvFunc : Set inversion function in domain.
SetInvTFunc : Set inversion test function in domain.
setjmp : Set jump environment.
SetLcmFunc : Set lcm function in domain.
SetMinus : Set minus.
SetMinusC : Set minus constructive.
SetMinusCQ : Set minus constructive equal.
SetMinusQ : Set minus equal.
SetNegFunc : Set negation function in domain.
SetOneFunc : Set one test function in domain.
SetPCppFunc : Set Content and primitive part function.
SetPFactFunc : Set factorization function in domain.
SetPNormFunc : Set polynomial normalform function in domain.
SetProdFunc : Set product function in domain.
SetPSpolFunc : Set polynomial S-polynomial function in domain.
SetPSqfrFunc : Set polynomial squarefree factorization function in domain.
SetPSugNormFunc : Set polynomial normal with sugar strategy normalform function in domain.
SetPSugSpolFunc : Set polynomial normal with sugar strategy S-polynomial function in domain.
SetQrFunc : Set quotient and remainder function in domain.
SetQuotFunc : Set quotient function in domain.
SetReadFunc : Set read function in domain.
SetReduceExp : Set Reduce-Exponent for procedure GroebnerBases2:
SetRemFunc : Set remainder function in domain.
SetRep : Set representation.
SetSignFunc : Set sign function in domain.

SetSumFunc : Set sum function in domain.
SetToipFunc : Set conversion-to-integer-polynomial function in domain.
SetTraceLevel : Set Trace-Level for decompositional groebner bases:
SetUnion : Set union.
SetUnion : Set union.
SetUnionQ : Set union equal.
SetUpdate : Set the update procedure.
SetUpdateProc : Set Update-Procedure for decompositional groebner bases:
SetUpdateVariableWeight : Set the DIPAGB variable weight update procedure.
SETV : Set variable.
SetVarOrdOpt : Set Variable-Order-Optimization for decompositional groebner bases:
SetVlddFunc : Set variable list from domain descriptor function in domain.
SetWritFunc : Set write function in domain.
SFCS : Set from characteristic set.
SFCS : Set from characteristic set.
SFIRST : Set first.
signal : Set system signal handler.
sigsetmask : Set signal mask.
SILINE : Set input line.
SINTER : Set intersection.
SIUNIT : Set input unit.
SLELT : Set list element.
SOLINE : Set output line.
SOUNIT : Set output unit.
SRED : Set reductum.
SUNION : Set union.
SUNIT1 : UGB set input unit 1.
SUNIT2 : UGB set input unit 2.
USCOMP : Unordered set composition.
USDIFF : Unordered set difference.
USETCT : Unordered set containment test.
USINT : Unordered set intersection.
USUN : Unordered set union.
ValisSet : Set valis.

setjmp

setjmp : Setjmp Foreign Module.

sets

ValisReset : Sets.

shift

APSHFT : Arbitrary precision floating point shift.
tfshiftaf : type formula shift atomic formula.
TfShiftVars : type formula shift variables.

shifted

ISSUM : Integer shifted sum.

shut

SHUT : Shut.

sign

ADSIGN : Arbitrary domain sign.
AFSIGN : Algebraic number field sign.
AFUPSR : Algebraic number field univariate polynomial, sign at a rational
APSIGN : Arbitrary precision floating point sign.
DIGBSI : Distributive polynomial system algebraic number G basis sign.
DIIPSG : Distributive integral polynomial sign.
DIRPSG : Distributive rational polynomial sign.
EIVSIG : Exterior integral vector sign.
EVDFSI : Exponent vector difference and sign.
IPSCPP : Integral polynomial sign, content, and primitive part.
IPSIGN : Integral polynomial sign.
ISIGNF : Integer sign function.
IUPBES : Integral univariate polynomial binary rational evaluation of sign.
MASSIGN : Sign.
RFSIGN : Rational function sign.
RNSIGN : Rational number sign.
RPBLGS : Rational polynomial base coefficients least common multiple, greatest common divisor, and sign.
RPSIGN : Rational polynomial sign.
SetSignFunc : Set sign function in domain.
TfSignChs : type formula sign changes.

signal

massig : MAS Signal Handling Foreign Module.
MASSIGNAL : MAS Signal Handling Definition Module.
raise : Raise signal s.
SigMask : Signal mask.
signal : Set system signal handler.
sigsetmask : Set signal mask.
SigUsr1HandleDefault : SIGUSR1 default signal handler.

signals

sigblock : Block signals.

signature

ADSIG : Arbitrary domain signature.
ISIG : Integral matrix signature.
Signature : Signature of a compiled function or procedure.

signum

EVSIGN : Exponent vector signum.

SIGUSR1

SigUsr1HandleDefault : SIGUSR1 default signal handler.

similiar

IPSRP : Integral polynomial similiar to rational polynomial.

simplification

PQCnfSimplify : polynomial equation cnf based simplification.
PQDnfSimplify : polynomial equation dnf based simplification.
pqmkaf : polynomial equation simplification make atomic formula.
pqpaf : polynomial equation simplification parse atomic formula.
pqprtaf : polynomial equation simplification print atomic formula.
PQSCNF : polynomial equation simplification normal form.
PQSDNF : polynomial equation simplification normal form.
pqsmart : polynomial equation atomic formula smart simplification.
PQSMPL : Polynomial Equation Simplification Definition Module.

simplify

FORSIMPLIFY : formula simplify.
FORSIMPLIFYP : formula simplify prune.
FORSMPL : formula simplify.
MLDSMPL : maslog demonstration simplify.
MLPQSMPL : Masload Polynomial Equation Simplify Definition Module.
PQSIMPLIFY : polynomial equation simplify.
pqsimplifyaf : polynomial equation simplify atomic formula.
PQSIMPLIFYP : polynomial equation simplify.
PQSMPL : polynomial equation simplify.
SimplifyNf : simplify normal form.

single

FRLSM : Fermat residue list, single modulus.
MPIQHS : Modular polynomial mod ideal, quadratic Hensel lemma on a single variable.

sinus

SIN : Sinus.

situation

WRS1 : Write Situation.
WRS2 : Write Situation.

SK

NOEL32 : Noether SK Polynomial Computation.
NOEPOW : Noether SK Power Sum Computation.
SUBPOW : Noether SK Power Sum Computation for Substitution Groups.

skipping

CREADB : Character read, skipping blanks.

small

ISPD : Integer small prime divisors.

smart

pqsmart : polynomial equation atomic formula smart simplification.

solution

IEQ : Special Solution for inhomogenous commutative equation.
IMUNS : Integer matrix upper triangular matrix solution null space.
ISEQ : Special Solution for inhomogenous commutative system of equation.
LDSMKB : Linear diophantine system solution, modified Kannan and Bachem algorithm.
LDSSBR : Linear diophantine system solution, based on Rosser ideas.
MIPISE : Modular integral polynomial mod ideal, solution of equation.
MIUPSE : Modular integral univariate polynomial, solution of equation.
NLIEQ : Special Solution for inhomogenous non-commutative equation.
NLISEQ : Special Solution for inhomogenous non-commutative system of equation.
RNMUNS : Rational number matrix upper triangular matrix solution null space.
SIC : Special Solution for inhomogenous commutative system of equation.
SINL : Special Solution for inhomogenous non-commutative system of equation.
STIC : Solution Test for inhomogenous commutative Case.
STINL : Solution Test for inhomogenous non-commutative Case.

solve

IMSDS : Integer matrix solve decomposed system.
RNMSDS : Rational number matrix solve decomposed system.
RRCSR : Real root count solve and reduce.

sort

DIDPCPLMS1 : Distributive domain polynomial list construct pairs list merge sort.
DIDPLM1 : Distributive domain polynomial list merge sort.
DIIPCPLMS1 : Distributive integral polynomial list construct pairs list merge sort.
DIIPLM1 : Distributive integral polynomial list merge sort.
DIIPSO : Distributive integral polynomial sort.
DILBBS : Distributive List Bubble Sort.
DILBSO : Distributive polynomial list bubble sort.
DILEBBS : Distributive List Extended Bubble Sort.
DIPBSO : Distributive polynomial bubble sort.
DIPLPM : Distributive polynomial list pair-merge sort.
DIPSSM : Distributive polynomial sort and select minimal.
DIRPSO : Distributive rational polynomial sort.
EVLRNBSO : Rational exponent vector list bubble sort.
EVPLSO : Exponent vector pair-list sort.
EVUMSORT : Exponent vector unique merge sort.
LBIBMS : List of beta-integers bubble-merge sort.
LBIBS : List of beta-integers bubble sort.
LRNBMS : List of rational numbers bubble-merge sort.
LRNBS : List of rational numbers bubble sort.

MICS : Matrix of integers column sort.
NOESRT : Noether Polynomial Sort.
REDSRT : Red terms sort.
WHSRT : White sort.

sorted

EVSSPROD : Exponent vektor set sorted product.
GS1 : UGB generate stack of sorted polynomials and critical pairs 1.
GS2 : UGB generate stack of sorted polynomials and critical pairs 2.

space

IMUNS : Integer matrix upper triangular matrix solution null space.
MMDNSB : Matrix of modular digits null space basis.
RNMUNS : Rational number matrix upper triangular matrix solution null space.

sparse

DIIRAS : Distributive integral polynomial random sparse exponent vector.
DIRRAS : Distributive rational polynomial, random sparse exponent vector.

special

GSYNSP : G-Symmetric Number of Special Polynomials.
IEQ : Special Solution for inhomogenous commutative equation.
IPRCHS : Integral polynomial real root calculation, high-precision special.
ISEQ : Special Solution for inhomogenous commutative system of equation.
MUPFS : Modular univariate polynomial factorization, special.
NLIEQ : Special Solution for inhomogenous non-commutative equation.
NLISEQ : Special Solution for inhomogenous non-commutative system of equation.
PQBASE : symbol to mark a polynomial equation special atomic formula.
PSDSV : Polynomial special decomposition, specified variable.
RNDWRS : Rational number decimal write special.
SIC : Special Solution for inhomogenous commutative system of equation.
SINL : Special Solution for inhomogenous non-commutative system of equation.
SPECIALFORM : Test if expression S is a special form.

specification

DSPEC : Define specification.
MASSPEC : MAS Specification Definition Module.

specified

DIPTCS : Distributive polynomial trailing coefficient specified variable.
IPSR : Integral polynomial specified roots.
PDEGSV : Polynomial degree, specified variable.
PSDSV : Polynomial special decomposition, specified variable.

split

DITSPL : DIP zero set tupel split.

spolynom

SPOL : Parametric spolynom.

sqrt

SQRT : Sqrt.

square

DSQRTF : Digit square root function.
ISQRT : Integer square root.

squarefree

ADPSFF : Arbitrary domain polynomial squarefree factorization.
AFSUPB : Algebraic number field squarefree univariate polynomial squarefree
AFSUPB : Algebraic number field squarefree univariate polynomial squarefree
AFUPBA : Algebraic number field univariate polynomial squarefree basis
AFUPCB : Algebraic number field univariate polynomial coarsest squarefree basis.
AFUPGS : Algebraic number field polynomial greatest squarefree divisor.
AFUPSF : Algebraic number field univariate polynomial squarefree factorization.
DIPSFF : distributive polynomial squarefree factorization.
IPCSFB : Integral polynomial coarsest squarefree basis.
IPFSFB : Integral polynomial finest squarefree basis.
IPPGSD : Integral polynomial primitive greatest squarefree divisor.
IPSF : Integral polynomial squarefree factorization.
IPSFBA : Integral polynomial squarefree basis augmentation.
IPSFF : integral polynomial squarefree factorization
IPSFSD : Integral squarefree factorization, second derivative.
ISFPF : Integral squarefree polynomial factorization.
ISFPIR : Integral squarefree polynomial isolating interval refinement.
ISPSFB : Integral squarefree polynomial squarefree basis.
ISPSFB : Integral squarefree polynomial squarefree basis.
IUSFPF : Integral univariate squarefree polynomial factorization.
MUPSFF : Modular univariate polynomial squarefree factorization.
SetPSqfrFunc : Set polynomial squarefree factorization function in domain.

stack

ALLELN : UGB all linear forms from stack of projections.
ALLLF : UGB all linear forms from stack of projections and print.
getstck : Get contents of stack register.
GS1 : UGB generate stack of sorted polynomials and critical pairs 1.
GS2 : UGB generate stack of sorted polynomials and critical pairs 2.
LFALL : UGB all linear forms from stack of projections 1.

stacks

MERGE : UGB merge stacks.

standard

DIIPWV : Distributive integral polynomial write with standard variable list.
DIRPWV : Distributive rational polynomial write with standard variable
IPRIMS : Integral polynomial real root isolation, modified Uspensky method, standard interval.
IPSIFI : Integral polynomial standard isolating interval from isolating interval.
IPSRMS : Integral polynomial strong real root isolation, modified Uspensky method, standard interval.
IUPVSI : Integral univariate polynomial, variations for standard interval.
STVL : Standard variable list.

start

PatternAStart : pattern and start.
SysInfoStart : system information start.

starting

EVTSZ : Exponent vector test if starting with i zero exponents.

step

StepRep : Step through representation.

stop

SysInfoStop : system information stop.

storage

MASSTOR : MAS Storage Definition Module.

store

FORVTSTORE : formula variable table store.

strategy

ADPSUGNF : Arbitrary domain polynomial normal with sugar strategy normalform.
ADPSUGSP : Arbitrary domain polynomial normal with sugar strategy S-polynomial.
EDIPSUGCON : Extended distributive polynomial normal with sugar strategy constructor.
EDIPSUGNOR : Extended distributive polynomial normal with sugar strategy normal form.
EDIPSUGSP : Extended distributive polynomial normal with sugar strategy S-polynomial.
SetDIPAGBOptions : Set the trace flag, the strategy and the variable weight list of the
SetDIPAGBStrategy : Set the DIPAGB strategy option for the extended critical pair selection.
SetDIPAGBVariableWeights : Set the DIPAGB variable weight list for the normal with sugar strategy.
SetPSugNormFunc : Set polynomial normal with sugar strategy normalform function in domain.

SetPSugSpolFunc : Set polynomial normal with sugar strategy S-polynomial function in domain.
WriteDIPAGBOptions : Write the current trace flag, strategy and variable weight list of the
WriteDIPAGBStrategy : Write the DIPAGB strategy option for the extended critical pair selection

stream

CLTIS : Character list to input stream.
EStreamKind : Error stream kind.
IStreamKind : Input stream kind.
OStreamKind : Output stream kind.
Summary : Summary of stream IO.
WriteDIPAGBTraceFlag: Write the DIPAGB trace flag in the output stream.
WriteDIPAGBVariableWeight: Write the DIPAGB variable weight list in the output stream.

strict

DINPTslT : Distributive polynomial non-commutative product table strict lex test.

string

ADDDFSTR : arbitrary domain domain descriptor from string.
LISTS : List from string.
SLIST : String from list.
SWRITE : String write.

strong

DINTSS : DIP normalized tupel strong separation.
ILSCMP : Index list strong compare.
IPSRM : Integral polynomial strong real root isolation, modified Uspensky method.
IPSRMS : Integral polynomial strong real root isolation, modified Uspensky method, standard interval.

structure

RRADSTRCONST : Real root arbitrary domain structure constants.
RRISTRCONST : Real root integral structure constants.
RRUADSTRCONST : Real root univariate arbitrary domain structure constants.
RRUISTRCONST : Real root univariate integral structure constants.

subresultant

IPPSC : Integral polynomial principal subresultant coefficients.
IPSPRS : Integral polynomial subresultant polynomial remainder sequence.
MPSPRS : Modular polynomial subresultant polynomial remainder sequence.

subroutine

IPICS : Integral polynomial integer content subroutine.
MPUCS : Modular polynomial univariate content subroutine.
STBALS : Symbol tree balance subroutine.

subset

CSSUB : Characteristic set subset.
IXSUBS : Indexed subset.

substitute

DIPADS : Distributive polynomial advance and substitute.
FORSUBSTVAR : formula substitute variable.
MLDSUBSTVAR : maslog demonstration substitute variables.

substitution

DIIPSU : Distributive integral polynomial substitution.
DIIPSV : Distributive integral polynomial substitution for main variable.
DIRPSU : Distributive rational polynomial substitution.
DIRPSV : Distributive rational polynomial substitution for main variable.
EVSU : Exponent vector substitution.
IPGSUB : Integral polynomial general substitution.
IPSMV : Integral polynomial substitution for main variable.
IPSUB : Integral polynomial substitution.
SUBINF : Substitution Group Polynomial System Information.
SUBLIS : Substitution with list.
SUBLIS : Substitution with list.
SUBORD : Substitution Group Order.
SUBORP : Substitution Group Orbit Polynomial.
SUBPOW : Noether SK Power Sum Computation for Substitution Groups.
SUBRED : Noether G-Symmetric Polynomial Computation for Substitution Groups.
SUBSGR : Substitution Group Read.
SUBSGW : Substitution Group Write.
SUBST : Substitution Group Polynomial System Definition Module.

subtract

EVCSUB : Exponent vector component subtract.

successful

DIGFET : DIP G base successful extension test.

suffix

SUFFIX : Suffix.

sugar

ADPSUGNF : Arbitrary domain polynomial normal with sugar strategy normalform.
ADPSUGSP : Arbitrary domain polynomial normal with sugar strategy S-polynomial.
ECPSUGAR : Extended critical pair select sugar.
EDIIFSUGNF : Extended distributive integral function polynomial normal with sugar
EDIIFSUGSP : Extended distributive integral function polynomial normal with sugar
EDIPSUGAR : Extended distributive polynomial sugar.
EDIPSUGCON : Extended distributive polynomial normal with sugar strategy constructor.
EDIPSUGNOR : Extended distributive polynomial normal with sugar strategy normal form.
EDIPSUGSP : Extended distributive polynomial normal with sugar strategy S-polynomial.
SetDIPAGBVariableWeight : Set the DIPAGB variable weight list for the normal with sugar strategy.
SetPSugNormFunc : Set polynomial normal with sugar strategy normalform function in domain.
SetPSugSpolFunc : Set polynomial normal with sugar strategy S-polynomial function in domain.

sum

ADMSUM : Arbitrary domain matrix sum.
ADSUM : Arbitrary domain sum.
ADVVSUM : Arbitrary domain vector vector sum.
AFPSUM : Algebraic number field polynomial sum.
AFSUM : Algebraic number field element sum.
APSUM : Arbitrary precision floating point sum.
CSUM : Complex number sum.
DIIPLS : Distributive integral polynomial list sum.
DIIPSM : Distributive integral polynomial sum.
DIIPSN : Distributive integral polynomial sum norm.
DILSUM : Distributive polynomial list sum.
DIPSUM : Distributive polynomial sum.
DIRPLS : Distributive rational polynomial list sum.
DIRPSM : Distributive rational polynomial sum.
DIRPSN : Distributive rational polynomial sum norm.
DMPSUM : Dense modular polynomial sum.
EIVSUM : Exterior integral vector sum.
EVSUM : Exponent vector sum.
FFSUM : Finite field sum.
IBCPS : Integer binomial coefficient partial sum.
IMSUM : Integer matrix sum.
IPSUM : Integral polynomial sum.
IPSUMN : Integral polynomial sum norm.
ISSUM : Integer shifted sum.
ISUM : Integer sum.
IVSSUM : Integer vector scalar sum.
IVSVSUM : Integer vector scalar and vector sum.
IVVSUM : Integer vector vector sum.
MDSUM : Modular digit sum.
MIPSUM : Modular integral polynomial sum.

MISUM : Modular integer sum.
MPSUM : Modular polynomial sum.
NOEPOW : Noether SK Power Sum Computation.
NOEPSM : Noether Polynomial Sum.
OSUM : Octonion number sum.
QSUM : Quaternion number sum.
RFSUM : Rational function sum.
RNMSUM : Rational number matrix sum.
RNSUM : Rational number sum.
RNVSSUM : Rational number vector scalar sum.
RNVSVSUM : Rational number vector scalar sum.
RNVVSUM : Rational number vector vector sum.
RPSUM : Rational polynomial sum.
RRIPQSUM : Real root integral polynomial quotient sum.
SetSumFunc : Set sum function in domain.
SFP : UGB distributive rational polynomial sum.
SUBPOW : Noether SK Power Sum Computation for Substitution Groups.
SysInfoSum : system information sum.
VISUM : Vector of integers sum.

summary

CompSummary : Compiled function and procedure summary.
DomSummary : Arbitrary domain summary.
StorSummary : MASSTOR Summary.
Summary : Summary of stream IO.
SymSummary : Summary of symbol system.
SymSummary : Summary of symbol system.

sumset

PARTSS : Partition sumset.

switch

SwitchParse : Switch parsing between generic / non-generic parse.

SWRITE

DLSWRITE : debug level SWRITE.

symbol

Class2Sym : classification to symbol.
DIP2SYM : Distributive polynomial to symbol term.
ENTER : Enter into symbol table.
ENTER : Enter into symbol table.
EXPLOD : Explode symbol.
EXPLOD : Explode symbol.
GENARRAY : Generate array reference symbol.
GENSYM : Generate symbol.
GENSYM : Generate symbol.
IJACS : Integer Jacobi symbol algorithm.
MASSYM : MAS Symbol Definition Module.
MASSYM2 : MAS/SAC Symbol System Definition Module 2.

symbols

symm

symmetric

DIRPES : Distributive rational polynomial elementary symmetric functions.
DIRPSR : Distributive rational polynomial symmetric function reduction.
GSYSPG : Symmetric Permutation Group.
MASLOADG : MAS Load Symmetric Functions Definition Module.
MIPFSM : Modular integral polynomial from symmetric modular.
POWSEV : Power of variable symmetric product with exterior vector.
SMFMI : Symmetric modular from modular integer.
SMFMIP : Symmetric modular from modular integral polynomial.
SUBSYM : G-symmetric Polynomial Symmetric Check.
SYMMFU : Symmetric Functions Definition Module.
UIPSIL : Univariate integral polynomial symmetric product with exterior index list.
UIPSIV : Univariate integral polynomial symmetric product with exterior integral vector.

syntax

MWRIT1 : Output in modula like syntax.
MWRITE : Output in modula like syntax.

system

CgbCd : Groebner system initial case distinction.
CgbCons : Groebner system construct.
CGBFGSYS : Comprehensive Groebner basis from Groebner system.
CGBSYS : Comprehensive-Groebner-Bases System Definition Module.
DIGBSI : Distributive polynomial system algebraic number G basis sign.
DIIPCOM : Distributive integral polynomial complete system.
DIITNT : Distributive polynomial system interval tupel from norm tupel.
DIITWR : Distributive polynomial system interval tupels write.
DINTWR : Distributive polynomial system normalized tupels write.
DINTZS : DIP nomalized tupels from system zero.
DIPC : DIP Common Polynomial System Definition Module.
DIPCJ : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPDEC0 : DIP Ideal Decomposition 0 System Definition Module.
DIPIB : DIP Common Polynomial System Definition Module in the sense of Janet.
DIPIDEAL : DIP Ideal System Definition Module.
DIPIIB : DIP Integral Polynomial System Definition Module in the sense of Janet.
DIPROOT : DIP Ideal Real Root System Definition Module.
DIROWR : Distributive polynomial system real root write.
DIRPCOM : Distributive rational polynom complete system.
GBSYS : Groebner system.
GBSYSF : Groebner system with factorization.
GsCd : Groebner system initial case distinction.
GsCons : Groebner system construct.
GsParts : Groebner system parts.
GsS : Groebner system system part.
GsS : Groebner system system part.
GsVd : Groebner system variable list and domain descriptor.

GsWrite : Groebner system write.
GSYINF : G-Symmetric Polynomial System Information.
GSYMFUIN : G-Symmetric Integral Polynomial System Definition Module.
GSYMFURN : G-Symmetric Rational Polynomial System Definition Module.
GSYS : Groebner system.
GSYSDIM : Groebner system dimension.
GSYSF : Groebner system with factorization.
GSYSRED : Reduce Groebner system.
HSEQ : Homogenous System of Equation.
IMSDS : Integer matrix solve decomposed system.
ISEQ : Special Solution for inhomogenous commutative system of equation.
LDSMKB : Linear diophantine system solution, modified Kannan and Bachem algorithm.
LDSSBR : Linear diophantine system solution, based on Rosser ideas.
MASBIOS : MAS Basic I/O System Definition Module.
MASCOMB : MAS Combinatorial System Definition Module.
MASPGCD : MAS Polynomial GCD and RES System Definition Module.
MASSYM2 : MAS/SAC Symbol System Definition Module 2.
MLOGIO : Maslog Input Output System Definition Module.
NLHSEQ : Non-Commutative Homogenous System of Equation.
NLISEQ : Special Solution for inhomogenous non-commutative system of equation.
NOEINF : Noether Polynomial System Information.
NOETHER : Noether Polynomial System Definition Module.
RNMSDS : Rational number matrix solve decomposed system.
SACBIOS : SAC Basic I/O System Definition Module.
SACCOMB : SAC Combinatorical System Definition Module.
SACIPOL : SAC Integer Polynomial System Definition Module.
SACLDIO : SAC Linear Diophantine Equation System Definition Module.
SACPGCD : SAC Polynomial GCD and RES System Definition Module.
SACPOL : SAC Polynomial System Definition Module.
SACSYM : SAC Symbol System Definition Module.
SDR : System of distinct representatives.
SIC : Special Solution for inhomogenous commutative system of equation.
signal : Set system signal handler.
SINL : Special Solution for inhomogenous non-commutative system of equation.
SSYTBAL : System symbol tree balance.
SUBINF : Substitution Group Polynomial System Information.
SUBST : Substitution Group Polynomial System Definition Module.
SYHC : Syzygy for homogenous commutative system of equation.
SYHNL : Syzygy for homogenous non-commutative system of equation.
SymSummary : Summary of symbol system.
SymSummary : Summary of symbol system.
SYSINFO : System Informations Definition Module.
SysInfoStart : system information start.
SysInfoStop : system information stop.
SysInfoSum : system information sum.
SysInfoWrite : system information write.

systems

RDSYS : Read polynomial systems.

syzygy

MASLOADS : MAS Load Syzygy Definition Module.
NLSYONP : Syzygy for one Polynomial.
SYGB : Syzygy for Groebner Base.
SYGBE : Syzygy for Groebner Base with Exponent Vector.
SYHC : Syzygy for homogenous commutative system of equation.
SYHNL : Syzygy for homogenous non-commutative system of equation.
SYONP : Syzygy for old Polynomials by new Polynomials.
SYTHC : Syzygy Test for homogenous commutative Case.
SYTHNL : Syzygy Test for homogenous non-commutative Case.
SYZFUNC : Syzygy Functions Definition Module.
SYZGB : Syzygy Groebner Base Definition Module.
SYZHLP : Syzygy Utility Programs Definition Module.
SYZMAIN : Syzygy Main Programs Definition Module.

table

DINPTL : Distributive polynomial non-commutative product table lookup.
DINPTslT : Distributive polynomial non-commutative product table strict lex test.
DINPTU : Distributive polynomial non-commutative product table update.
ENTER : Enter into symbol table.
ENTER : Enter into symbol table.
FORVTENTER : formula variable table enter.
FORVTGET : formula variable table get.
FORVTRESTORE : formula variable table restore.
FORVTSTORE : formula variable table store.
STCNT : Symbol table tree count.
STCNT : Symbol table tree count.

tabulate

TAB : Tabulate.

tag

TAG : Tag object.

tagged

DECOFTAG : Descriptor of tagged object.
TYPOFTAG : Type of tagged object.
VALOFTAG : Value of tagged object.

tangens

ARCTAN : Arcus tangens.
TAN : Tangens.

tell

InitExternalsG : Tell Modula and LISP about external compiled procedures.
InitExternalsS : Tell Modula and LISP about external compiled procedures.

tensor

IMRTPROD : Integral matrix right tensor product.

term

ADEPleadingterm : Arbitrary polynomial leading term.
ColHT : Colouring head term.
ColpHT : Coloured polynomial head term.
DIP2SYM : Distributive polynomial to symbol term.
DIPCLT : Distributiv Polynomial Class of Term.
DIPTODEF : DIP define distributive polynomial term order.
DIRPFT : Distributive rational polynomial from term.
GSYADD : G-Symmetric Term Adder.
GSYTWG : G-Symmetric Term Weight.
SYM2DIP : Symbol term to distributive polynomial.
TESTHT : Test highest term.
TFDIRP : Term from distributive rational polynomial.
TVARS : Term variables.
WMEMB : White term member.
WRTERM : Write term.

termorder

DIPTOO : DIP Termorder Optimization Definition Module.

terms

DIPTRM : Distributive polynomial terms.
ECPLCMHT : Extended critical pair select the least common multiple of head terms.
GSYMLT : G-Symmetric Multilinear Terms.
ISNEU : UGB new terms test.
NEULF : UGB compute new linear forms from new terms.
NEWL : UGB update linear forms from new terms.
NONEWL : UGB update linear forms without new terms.
PTERM : Polynomial terms.
REDSRT : Red terms sort.
RFRED : Rational function reduction to lowest terms.
RMGRT : Remove green terms.
RNRED : Rational number reduction to lowest terms.
RRREDTERMS : Real root reduced terms.

test

ADCNST : Arbitrary domain constant test.
ADINVT : Arbitrary domain inverse existence test.
DIGBZT : Distributive polynomial groebner base common zero test.
DIGFET : DIP G base successful extension test.
DINPTslT : Distributive polynomial non-commutative product table strict lex test.
DIPUNT : Distributive polynomial univariate test.
DIRLCT : Distributive rational polynomial list ideal containment test.
EVGBIT : Exponent vector groebner base intersection test.
EVMT : Exponent vector multiple test.
EVMTJ : Exponent vector multiple test in the sense of Janet.
EVT : Exponent Vector Test.

EVTSZ : Exponent vector test if starting with i zero exponents.
FORTST : formula test.
GBHELP : Groebner test help.
IdealMember : ideal membership test.
ISNEU : UGB new terms test.
ISNEUL : UGB new linear form test.
ISPT : Integer selfridge primality test.
LAMBDAP : Test if expression S is a lambda form.
MEMBER : Membership test.
MEMQ : Membership test equal pointers.
MLDTST : maslog demonstration test 1.
NULRNV : Rational number vector null test.
OCCURQ : Occurs test equal pointers.
PUNT : Polynomial univariate test.
RadicalMember : radical membership test.
RRREDTEST : Real root reducibility test.
RRZDIM : Real root zero-dimensional test.
SetCnstFunc : Set constant test function in domain.
SetInvTFunc : Set inversion test function in domain.
SetOneFunc : Set one test function in domain.
SEXPRP : Test if X is a S-expression function.
SPECIALFORM : Test if expression S is a special form.
STIC : Solution Test for inhomogenous commutative Case.
STINL : Solution Test for inhomogenous non-commutative Case.
SYTHC : Syzygy Test for homogenous commutative Case.
SYTHNL : Syzygy Test for homogenous non-commutative Case.
TESTHT : Test highest term.
USETCT : Unordered set containment test.
WRTEST : Write groebner test.

tex

FORTEXW : formula tex write.
FORTEXWLVAR : tex write list of varaiables.
FORTEXWVAR : formula tex write variable.
MLDTEXW : maslog demonstration tex write.
PQTEXW : polynomial equation tex write.
pqtexwaf : polynomia equation tex write atomic formula.

TFGEN

TFGENI : TFGEN interpreter version.

tfZeroes

TfZeroesI : TfZeroes interpreter version.

the

INITUPDATE : The initialization function as a first call of UPDATE.

third

THIRD : Third.

trace

ADMPTRACE : Arbitrary domain matrix product trace.
ADMTRACE : Arbitrary domain matrix trace.
COMPA1 : UGB trace member in trace list.
COMPA1 : UGB trace member in trace list.
COMPA2 : UGB trace compare.
CSPUR : UGB trace for linear form 2.
IMPTRACE : Integral matrix product trace.
IMTRACE : Integral matrix trace.
MKLIST : UGB make trace and cuts.
PKEGEL : UGB trace for linear form.
SetDCIBTraceLevel : Set Decompositional involutive base Trace Level.
SetDIPAGBOptions : Set the trace flag, the strategy and the variable weight list of the
SetDIPAGBTraceFlag : Set the DIPAGB trace flag.
SetDIPIBTraceLevel : Set Trace Level.
WriteDIPAGBOptions : Write the current trace flag, strategy and variable weight list of the
WriteDIPAGBTraceFlag: Write the DIPAGB trace flag in the output stream.

trace-Level

SetTraceLevel : Set Trace-Level for decompositional groebner bases:

trailing

DIPTBC : Distributive polynomial trailing base coefficient.
DIPTCF : Distributive polynomial trailing coefficient.
DIPTCS : Distributive polynomial trailing coefficient specified variable.
ITD : Integer trailing digit.
PTBCF : Polynomial trailing base coefficient.

transformation

IMLT : Integer lower triangular matrix transformation.
IMUT : Integer upper triangular matrix transformation.
IPVCHT : Integral polynomial variations after circle to half-plane transformation.
IUPBHT : Integral univariate polynomial binary homothetic transformation.
IUPCHT : Integral univariate polynomial circle to half-plane transformation.
IUPIHT : Integral univariate polynomial integer homothetic transformation.
IUPNT : Integral univariate polynomial negative transformation.
MINNCT : Matrix of integers, non-negative column transformation.
PRT : Polynomial reciprocal transformation.
RINT : Rational interval normalizing transformation.
RNMLT : Rational matrix lower triangular matrix transformation.
RNMUT : Rational matrix upper triangular matrix transformation.
VIUT : Vector of integers, unimodular transformation.

translation

DIIPTM : Distributive integral polynomial translation main variable.
DIIPTR : Distributive integral polynomial translation.
DIRPTM : Distributive rational polynomial translation main variable.
DIRPTR : Distributive rational polynomial translation.
IPTRAN : Integral polynomial translation.
IPTRMV : Integral polynomial translation, main variable.
IUPTR : Integral univariate polynomial translation.
IUPTR1 : Integral univariate polynomial translation by 1.

transpose

MTRANS : Matrix transpose.

tree

SetDCIBdepth : Set decompositional involutive base depth of tree.
SSYTBAL : System symbol tree balance.
STBAL : Symbol tree balance.
STBALS : Symbol tree balance subroutine.
STCNT : Symbol table tree count.
STCNT : Symbol table tree count.
STINS : Symbol tree insertion.
STINS : Symbol tree insertion.
STLST : Symbol tree list.
STLST : Symbol tree list.
STLSTI : Symbol tree list, in-order.
STLSTI : Symbol tree list, in-order.
STNLST : Symbol tree nodes list.
STSRCH : Symbol tree search.
STSRCH : Symbol tree search.
STWRT : Symbol tree write.
STWRT : Symbol tree write.

triangular

IMLT : Integer lower triangular matrix transformation.
IMUNS : Integer matrix upper triangular matrix solution null space.
IMUT : Integer upper triangular matrix transformation.
RNMLT : Rational matrix lower triangular matrix transformation.
RNMUNS : Rational number matrix upper triangular matrix solution null space.
RNMUT : Rational matrix upper triangular matrix transformation.

triple

ADEPtriple : Arbitrary domain extende polynomial triple.

truncated

IPTPR : Integral polynomial truncated product.
IUPTPR : Integral univariate polynomial truncated product.

truncation

IPTRUN : Integral polynomial truncation.
ITRUNC : Integer truncation.

tupel

DIITNT : Distributive polynomial system interval tupel from norm tupel.
DIITNT : Distributive polynomial system interval tupel from norm tupel.
DINTFE : DIP normalized tupel field extension.
DINTSR : DIP normalized tupel separation refinement.
DINTSS : DIP normalized tupel strong separation.
DITFZS : DIP tupel from zero set.
DITSPL : DIP zero set tupel split.

tupels

DIITWR : Distributive polynomial system interval tupels write.
DINTWR : Distributive polynomial system normalized tupels write.
DINTZS : DIP nomalized tupels from system zero.
RRVTEXT : Real root vector of tupels extension.

tuple

DIPCT : distributive polynomial coefficient tuple.
TfClassify : type formula classify coefficient tuple.
TfClassifyI : type formula classify coefficient tuple interpreter version.
TFFTUPLE : type formula from coefficient tuple with joker entries.
TfNextTuple : type formula next tuple.

tuples

TfCtj : type formula coefficient tuples with joker argument.

two

DIFF1 : UGB difference set for two rational exponent vector list.
POLCOP : Two level list copy.

typ

DIPTYP : Distributive polynomial typ.
PTYP : Polynomial typ.

type

ComputeTypeFormula : compute type formula.

TYPE

DIPDCGB : TYPE PROCF1 = PROCEDURE(LIST): LIST;

type

TfClassify : type formula classify coefficient tuple.
TfClassifyI : type formula classify coefficient tuple interpreter version.
TfComputeTf : type formula compute type formulas.
TfComputeTf : type formula compute type formulas.
TfCount : type formula count.
TfCount1 : type formula count 1.
TfCtj : type formula coefficient tuples with joker argument.
TFFTUPLE : type formula from coefficient tuple with joker entries.
TFGEN : type formula generate.
TFGENJ : type formula generate with joker argument.
tfgrel : type formula get relation symbol.
TFIREAD : type formula infix read.
tfmkaf : type formula make atomic formula.
TfNextTuple : type formula next tuple.
TFORM : Type Formula Definition Module.
tfpaf : type formula parse atomic formula.

TFPPRT : type formula pretty print.
TfRecBasis : type formula recursion basis.
tfshiftaf : type formula shift atomic formula.
TfShiftVars : type formula shift variables.
TfSignChs : type formula sign changes.
TfTypeFormula : type formula type formula.
TfTypeFormula : type formula type formula.
TfUseDb : type formula use data base.
TfZeroes : type formula zeroes.
TfZeroes0 : type formula zeroes 0.
TYPEOF : Type of S-expression.
TYPOFTAG : Type of tagged object.

typed

GENTE : Generate typed expression.

types

MASLISPU : Types, S-Expresion Types and Indicators.
MASLISPU : Types, S-Expresion Types and Indicators.

UGB

ALLELN : UGB all linear forms from stack of projections.
ALLLF : UGB all linear forms from stack of projections and print.
CLF2 : UGB compute linear form from difference set 2.
CLF3 : UGB compute linear form from difference set 3.
COMPA1 : UGB trace member in trace list.
COMPA2 : UGB trace compare.
COMPLF : UGB compute linear form from difference set.
CP2 : UGB linear form product with rational exponent vector list 2.
CQ2 : UGB linear form product with rational exponent vector list.
CSPUR : UGB trace for linear form 2.
CUT : UGB set of cuts.
DEGRE : UGB total degree of a list of rational exponent vectors.
DFP : UGB distributive rational polynomial difference.
DIFF : UGB difference set for rational exponent vector list.
DIFF1 : UGB difference set for two rational exponent vector list.
DIPMC2 : UGB distributive polynomial composition 2.
DIRPSP : UGB distributive polynomial S-polynomial.
DIRRNF : UGB distributive polynomial normalform.
DO1 : UGB add last component to exponent vector.
EVCOMP : UGB exponent vector compare.
EVLFCP : UGB exponent vector linear form compare.
EXECRD : UGB execution options read.
EXEUGB : UGB execute.
EXPTU : UGB extract exponent vector list from polynomial list.
GS1 : UGB generate stack of sorted polynomials and critical pairs 1.
GS2 : UGB generate stack of sorted polynomials and critical pairs 2.
ISNEU : UGB new terms test.
ISNEUL : UGB new linear form test.
LF : UGB linear form.
LFALL : UGB all linear forms from stack of projections 1.

LFGET : UGB get linear form from list of linear forms.
MAKERN : UGB rational exponent vector list from integer ev list.
MERGE : UGB merge stacks.
MKLF1 : UGB make new linear forms 1.
MKLF2 : UGB make new linear forms 2.
MKLF3 : UGB make new linear forms 3.
MKLIST : UGB make trace and cuts.
MKNEWP : UGB make new critical pairs.
MKPAIR : UGB make critical pairs for polynomial list.
MKSET : UGB rational exponent vector list difference list.
MKSP1 : UGB compute next non-zero reduced S-polynomial.
NEULF : UGB compute new linear forms from new terms.
NEWDIF : UGB exponent vector list difference from polynomials.
NEWL : UGB update linear forms from new terms.
NONEWL : UGB update linear forms without new terms.
OPREAD : UGB options and parameter read.
PCOMP : UGB distributive polynomial composition.
PDIF : UGB rational exponent vector list difference list, incremental.
PKEGEL : UGB trace for linear form.
PLF : UGB linear form with precomputed linear forms.
PREAD : UGB polynomial read.
PROJ : UGB projection, one dimension.
PROJEC : UGB projection to dimension 1.
RDPAR : UGB read parameter.
RNVDIF : UGB rational exponent vector difference.
SCMULT : UGB rational exponent vector rational number product.
SCPROD : UGB rational exponent vector scalar product.
SEENR : UGB number of option.
SFP : UGB distributive rational polynomial sum.
SKPRO2 : UGB rational exponent vector scalar product with integer ev.
SUNIT1 : UGB set input unit 1.
SUNIT2 : UGB set input unit 2.
TCOMP : UGB list constructive conc.
UGBBIN : UGB input, execute and output.
ZULFO : UGB find admissible extensions of linear forms.

un-extend

ECPUNEXTEND : Extended critical pair un-extend.
EDIPUNEXTEND : Extended distributive polynomial un-extend.
LECPUNEXTEND : List of extended critical pairs un-extend.
LEDIPUNEXTEND : List of extended distributive polynomials un-extend.

un-extension

SetEDIPUnExtend : Set the extended distributive polynomial un-extension function.

unary

FORMKUNOP : formula make unary operation.
FORPUNOP : formula parse unary operation.
FORPUNOPA : formula parse unary operation argument.

unification

UNIFY : Unification.

unimodular

VIUT : Vector of integers, unimodular transformation.

union

CSUN : Characteristic set union.
DIPADGBunion : distributive polynomial arbitrary domain groebner basis union.
SetUnion : Set union.
SetUnion : Set union.
SetUnionQ : Set union equal.
SUNION : Set union.
USUN : Unordered set union.

unique

EVUMSORT : Exponent vector unique merge sort.

unit

ADUM : Arbitrary domain unit matrix.
CUNIT : Close unit.
IPRIMU : Integral polynomial real root isolation, modified Uspensky method, unit interval.
IUM : Integer unit matrix.
RNUM : Rational number unit matrix.
SIUNIT : Set input unit.
SOUNIT : Set output unit.
SUNIT1 : UGB set input unit 1.
SUNIT2 : UGB set input unit 2.

univariate

AFSUPB : Algebraic number field squarefree univariate polynomial squarefree
AFUPBA : Algebraic number field univariate polynomial squarefree basis
AFUPCB : Algebraic number field univariate polynomial coarsest squarefree basis.
AFUPGC : Algebraic number field univariate polynomial greatest common divisor
AFUPRB : Algebraic number field univariate polynomial root bound.
AFUPSF : Algebraic number field univariate polynomial squarefree factorization.
AFUPSR : Algebraic number field univariate polynomial, sign at a rational
DIPUNT : Distributive polynomial univariate test.
DIPUV : Distributive polynomial univariate variable output.
DMUPNR : Dense modular univariate polynomial natural remainder.
EIVFUP : Exterior integral vector from univariate integral polynomial
IUPBEI : Integral univariate polynomial binary rational evaluation, integer output.
IUPBES : Integral univariate polynomial binary rational evaluation of sign.
IUPBHT : Integral univariate polynomial binary homothetic transformation.
IUPBRE : Integral univariate polynomial binary rational evaluation.
IUPCHT : Integral univariate polynomial circle to half-plane transformation.

IUPFAC : Integral univariate polynomial factorization.
IUPFDS : Integral univariate polynomial factor degree set.
IUPIHT : Integral univariate polynomial integer homothetic transformation.
IUPMRN : Integral univariate polynomial minimal polynomial of a rational number.
IUPNT : Integral univariate polynomial negative transformation.
IUPQH : Integral univariate polynomial quadratic Hensel lemma.
IUPQHL : Integral univariate polynomial quadratic Hensel lemma, list.
IUPRB : Integral univariate polynomial root bound.
IUPRC : Integral univariate polynomial resultant and cofactor.
IUPRLP : Integral univariate polynomial, root of a linear polynomial.
IUPTPR : Integral univariate polynomial truncated product.
IUPTR : Integral univariate polynomial translation.
IUPTR1 : Integral univariate polynomial translation by 1.
IUPVAR : Integral univariate polynomial variations.
IUPVOI : Integral univariate polynomial, variations for open interval.
IUPVSI : Integral univariate polynomial, variations for standard interval.
IUSFPF : Integral univariate squarefree polynomial factorization.
MIUPQR : Modular integral univariate polynomial quotient and remainder.
MIUPSE : Modular integral univariate polynomial, solution of equation.
MPUC : Modular polynomial univariate content.
MPUCPP : Modular polynomial univariate content and primitive part.
MPUCS : Modular polynomial univariate content subroutine.
MPUP : Modular polynomial univariate product.
MPUPP : Modular polynomial univariate primitive part.
MPUQ : Modular polynomial univariate quotient.
MUPBQP : Modular univariate polynomial Berlekamp q polynomials construction.
MUPDDF : Modular univariate polynomial distinct degree factorization.
MUPDER : Modular univariate polynomial derivative.
MUPEGC : Modular univariate polynomial extended greatest common divisor.
MUPFBL : Modular univariate polynomial factorization-Berlekamp algorithm.
MUPFS : Modular univariate polynomial factorization, special.
MUPGCD : Modular univariate polynomial greatest common divisor.
MUPHEG : Modular univariate polynomial half-extended greatest common divisor.
MUPRAN : Modular univariate polynomial, random.
MUPRC : Modular univariate polynomial resultant and cofactor.
MUPRES : Modular univariate polynomial resultant.
MUPSFF : Modular univariate polynomial squarefree factorization.
PUFP : Polynomial, univariate, from polynomial.
PUNT : Polynomial univariate test.
RRUADCOUNT : Real root univariate arbirary domain count.
RRUADOM : Real Root Univariate Arbitrary Domain Definition Module.
RRUADPOLTOVEC : Real root univariate arbitrary domain polynomial to vector.
RRUADQUADFORM : Real root univariate arbitrary domain quadratic form.
RRUADSTRCONST : Real root univariate arbitrary domain structure constants.
RRUICOUNT : Real root univariate integral count.
RRUINT : Real Root Univariate Integral Definition Module.
RRUIPOLTOVEC : Real root univariate integral polynomial to vector.
RRUIQUADFORM : Real root univariate integral quadratic form.
RRUISTRCONST : Real root univariate integral structure constants.
RUPEGC : Rational univariate polynomial extended greatest common divisor.
RUPGCD : Rational univariate polynomial greatest common divisor.

RUPHEG : Rational univariate polynomial half-extended greatest common divisor.
RUPLCM : Rational univariate polynomial least common multiple.
RUPMRN : Rational univariate polynomial minimal polynomial of a rational number.
SACMUFAC : SAC Modular Univariate Polynomial Factorization Definition Module.
SACUPFAC : SAC Univariate Polynomial Factorization Definition Module.
UIPRES : Univariate integral polynomials resultant.
UIPRS1 : Univariate integral polynomials resultant 1.
UIPSIL : Univariate integral polynomial symmetric product with exterior index list.
UIPSIV : Univariate integral polynomial symmetric product with exterior integral vector.

univariats

DIGISM : DIP G base index search for extension multiple univariats.

universal

MASUGB : Universal Groebner Bases Definition Module.
PUG : Universal Groebner base using precomputation.
PUGB : Universal Groebner base with precomputed linear forms.
UG : Universal Groebner base.
UGB : Universal Groebner base.
UREAD : Universal read.
UREAD : Universal read.
UREAD : Universal read.
UWRIT1 : Universal write, 1.
UWRIT1 : Universal write, 1.
UWRIT1 : Universal write, 1.
UWRITE : Universal write.
UWRITE : Universal write.
UWRITE : Universal write.
WRUGB : Write universal Groebner base.
WRUGF : Write universal Groebner family.

unordered

USCOMP : Unordered set composition.
USDIFF : Unordered set difference.
USETCT : Unordered set containment test.
USINT : Unordered set intersection.
USUN : Unordered set union.

until

BACKUB : Backspace until blank.

untriple

ADEPuntriple : Arbitrary domain extended polynomial untriple.

upcase

UPCASE : upcase character list.

update

BGFUP : Base Generators Factor Update.
DIDPUCPL1 : Distributive domain polynomial update constructed pairs list.
DILUPL : Distributive polynomial list update pair list.
DINPTU : Distributive polynomial non-commutative product table update.
DNLUPL : distributive polynomial non-noetherian left update pair list.
DNRUPL : distributive polynomial non-noetherian right update pair list.
GBUPD : Groebner-system update.
GSYSN0 : Groebner-System n0 update.

UPDATE

INITUPDATE : The initialization function as a first call of UPDATE.

update

NEWL : UGB update linear forms from new terms.
NLBGFUP : Non-Commutative Base Generators Factor Update.
NONEWL : UGB update linear forms without new terms.
SetUpdate : Set the update procedure.
SetUpdateVariableWeight : Set the DIPAGB variable weight update procedure.
UPDATE : Update of extended ideal basis and extended critical pair list as required
UpdateVariableWeight : Update of the DIPAGB variable weight list.
UPDCAS : Update case distinction.
UPDPP : Update polynomials.
WUPD : White part update.

update-Procedure

SetUpdateProc : Set Update-Procedure for decompositional groebner bases:

upper

IMUNS : Integer matrix upper triangular matrix solution null space.
IMUT : Integer upper triangular matrix transformation.
RNMUNS : Rational number matrix upper triangular matrix solution null space.
RNMUT : Rational matrix upper triangular matrix transformation.

use

TfUseDb : type formula use data base.
UseDb : Use data base.

using

DIILFRCD : DIP integral list from DIP rational list using common denominator.
IMDET : Integer matrix determinant, using Gaussian elimination.
IMDETL : Integer matrix determinant, using Laplace expansion.
PUG : Universal Groebner base using precomputation.
RNMDET : Rational number matrix determinant, using Gaussian elimination.
RNMDETL : Rational number matrix determinant, using Laplace expansion.

uspensky

IPRIM : Integral polynomial real root isolation, modified Uspensky method.
IPRIMO : Integral polynomial real root isolation, modified Uspensky method, open interval.
IPRIMS : Integral polynomial real root isolation, modified Uspensky method, standard interval.
IPRIMU : Integral polynomial real root isolation, modified Uspensky method, unit interval.
IPRIU : Integral polynomial real root isolation, Uspensky method.
IPRIUP : Integral polynomial real root isolation, Uspensky method, positive roots.
IPSRM : Integral polynomial strong real root isolation, modified Uspensky method.
IPSRMS : Integral polynomial strong real root isolation, modified Uspensky method, standard interval.

utility

CGBFUNC : Comprehensive-Groebner-Bases Utility Functions Definition Module.
InitExternalsU : Initialize external compiled utility procedures.
MASBIOSU : MAS BIOS Utility Definition Module.
MASLISPU : MAS Lisp Utility Definition Module.
MASU : MAS Utility Definition Module.
SYZHLP : Syzygy Utility Programs Definition Module.

valis

ValisPop : valis pop.
ValisPush : valis push.
ValisReset : Reset valis.
ValisSet : Set valis.

value

ADABSF : Arbitrary domain absolute value.
APABS : Arbitrary precision floating point absolute value.
CABS : Complex number absolute value.
DIIPAB : Distributive integral polynomial absolute value.
DIRPAB : Distributive rational polynomial absolute value.
EIVABS : Exterior integral vector absolute value.
IABSF : Integer absolute value function.
IPABS : Integral polynomial absolute value.
MASABS : Absolute value.
OABS : Octonion number absolute value.
QABS : Quaternion number absolute value.
RNABS : Rational number absolute value.
RPABS : Rational polynomial absolute value.
SetAbsFunc : Set absolute value function in domain.
VALOFTAG : Value of tagged object.

values

RNVABS : Rational number list absolute values.

varaiables

FORTEXWLVAR : tex write list of varaiables.

variable

ADFACTO : Arbitrary domain factorization with variable order optimization.
ADVLDD : variable list from domain descriptor.
AFPDMV : Algebraic number field polynomial derivative, main variable.
AFPEMV : Algebraic number field polynomial evaluation of main variable.
CdpVd : Case distinction and polynomial set variable list and domain descriptor
CgbVd : Comprehensive Groebner basis variable list and domain descriptor.
DIIPDM : Distributive integral polynomial derivation main variable.
DIIPEM : Distributive integral polynomial evaluation of main variable.
DIIPEV : Distributive integral polynomial evaluation of the i-th variable.
DIIPSV : Distributive integral polynomial substitution for main variable.
DIIPTM : Distributive integral polynomial translation main variable.
DIIPWV : Distributive integral polynomial write with standard variable list.
DILINV : distributive polynomial list introduce new variable.
DIPADM : Distributive polynomial advance main variable.
DIPDEGI : distributive polynomial degree of i-th main variable.
DIPDPV : Distributive polynomial division by power of variable.
DIPMPM : Distributive polynomial multiplication by power of main variable.
DIPMPV : Distributive polynomial multiplication by power of variable.
DIPMVV : distributive polynomial minimal variable vector.
DIPNML : Distributive polynomial nonmultiple variable list.
DIPTCS : Distributive polynomial trailing coefficient specified variable.
DIPUV : Distributive polynomial univariate variable output.
DIPVDEF : DIP define distributive polynomial variable list.
DIPVOP : Distributive polynomial variable ordering optimisation.
DIPVOPP : Distributive polynomial variable ordering optimization and permutation
DIRPDM : Distributive rational polynomial derivation main variable.
DIRPEM : Distributive rational polynomial evaluation of main variable.
DIRPEV : Distributive rational polynomial evaluation of the i-th variable.
DIRPMV : Distributiv Polynomial multiplication with a variable.
DIRPSV : Distributive rational polynomial substitution for main variable.
DIRPTM : Distributive rational polynomial translation main variable.
DIRPWV : Distributive rational polynomial write with standard variable
FdV : Formula and dimension variable list part.
FORCONTBDVAR : formula contain bound variable.
FORCONTVAR : formula contain variable.
FORISBOOLVAR : formula is boolean variable.
FORISLVAR : formula is variable list.
FORISVAR : formula is variable.
FORMKVAR : formula make variable.
FORMKVD : formula make variable names disjoint.
FORPPVAR : formula pretty print variable.
FORPVAR : formula parse variable.
FORPVARA : formula parse variable arguments.
FORRDVAR : formula read variable.
FORSUBSTVAR : formula substitute variable.
FORTEXWVAR : formula tex write variable.

VdV : Variable list and domain descriptor variable list part.
VdV : Variable list and domain descriptor variable list part.
vlistflvar : variable list from lvar.
VLREAD : Variable list read.
VLSRCH : Variable list search.
VLWRIT : Variable list write.
VREAD : Variable read.
VVECC : variable vector complement.
VVECFVLIST : variable vector from variable lists.
VVECFVLIST : variable vector from variable lists.
WriteDIPAGBOptions : Write the current trace flag, strategy and variable weight list of the
WriteDIPAGBVariableWeight : Write the DIPAGB variable weight list in the output stream.

variable-Order-Optimization

SetVarOrdOpt : Set Variable-Order-Optimization for decompositional groebner bases:

variables

DILPERM : distributive polynomial list permutation of variables.
DIPCNSTR : distributive polynomial constant relative to variables.
DIPERM : Distributive polynomial permutation of variables.
DIPEXC : Distributive polynomial exchange variables.
DIPINV : Distributive polynomial introduction of new variables.
DIPLDV : Distributive polynomial list dependency on variables.
DIPNOV : Distributive polynomial number of variables.
DIPVL : Distributive Polynomial List of Variables.
EVDOV : Exponent vector dependency on variables.
EVINV : Exponent vector introduction of new variables.
EVLINV : Exponent vector list introduction of new variables.
FORGLVAR : formula get list of variables.
FORMKLVAR : formula make list of variables.
FORPLVAR : formula parse list of variables.
FORRDLVAR : formula read list of variables.
MLDMKVD : maslog demonstration make variables disjoint.
MLDSUBSTVAR : maslog demonstration substitute variables.
PINV : Polynomial introduction of new variables.
PPERMV : Polynomial permutation of variables.
RFNOV : Rational function number of variables.
RRADVARMATRICES : Real root arbitrary domain multiplication matrices of variables.
RRIVARMATRICES : Real root integral multiplication matrices of variables.
TfShiftVars : type formula shift variables.
TVARS : Term variables.

variations

IPVCHT : Integral polynomial variations after circle to half-plane transformation.
IUPVAR : Integral univariate polynomial variations.
IUPVOI : Integral univariate polynomial, variations for open interval.
IUPVSI : Integral univariate polynomial, variations for standard interval.

vector

ADVSPROD : Arbitrary domain vector scalar product.
ADVSVPROD : Arbitrary domain vector scalar vector product.
ADVSVPROD : Arbitrary domain vector scalar vector product.
ADVVSUM : Arbitrary domain vector vector sum.
ADVVSUM : Arbitrary domain vector vector sum.
ADVWRITE : Arbitrary domain vector write.
CP2 : UGB linear form product with rational exponent vector list 2.
CQ2 : UGB linear form product with rational exponent vector list.
DIFF : UGB difference set for rational exponent vector list.
DIFF1 : UGB difference set for two rational exponent vector list.
DIIRAS : Distributive integral polynomial random sparse exponent vector.
DILFEL : Distributive polynomial list from exponent vector list.
DIPDEV : Distributive polynomial degree vector.
DIPEVL : Distributive polynomial exponent vector leading monomial.
DIPEVP : Distributive polynomial exponent vector product.
DIPMVV : distributive polynomial minimal variable vector.
DIRRAS : Distributive rational polynomial, random sparse exponent vector.
DMEVAD : Degree matrix exponent vector add.
DO1 : UGB add last component to exponent vector.
EDIPEVL : Extended distributive polynomial exponent vector of the leading monomial.
EIVABS : Exterior integral vector absolute value.
EIVAPP : Exterior integral vector absolute primitive part.
EIVCPP : Exterior integral vector content and primitive part.
EIVEPR : Exterior integral vector exterior product.
EIVFUP : Exterior integral vector from univariate integral polynomial
EIVILP : Exterior integral vector inner left product.
EIVIP : Exterior integral vector integer product.
EIVIQ : Exterior integral vector integer quotient.
EIVIRP : Exterior integral vector inner right product.
EIVNEG : Exterior integral vector negative.
EIVPP : Exterior integral vector primitive part.
EIVSIG : Exterior integral vector sign.
EIVSUM : Exterior integral vector sum.
EIVWRT : Exterior integral vector write.
EVASC : Exponent vector ascending.
EVCADD : Exponent vector component add.
EVCNSTR : exponent vector constant relatively.
EVCOMP : Exponent vector compare.
EVCOMP : UGB exponent vector compare.
EVCSUB : Exponent vector component subtract.
EVDEL : Exponent vector delete.
EVDER : Exponent vector derivation.
EVDFSI : Exponent vector difference and sign.
EVDIF : Exponent vector difference.
EVDIF2 : Exponent vector difference.
EVDOV : Exponent vector dependency on variables.
EVEXC : Exponent vector exchange.
EVEXT : exponent vector extension.
EVF : Exponent Vector First.
EVGBIT : Exponent vector groebner base intersection test.
EVGCD : Exponent vector greatest common divisor.

EVIGLC : Exponent vector inverse graded lexicographical compare.
EVILCI : Exponent vector inverse lexicographical compare inverse exponent vector.
EVILCI : Exponent vector inverse lexicographical compare inverse exponent vector.
EVILCP : Exponent vector inverse lexicographical compare.
EVINV : Exponent vector introduction of new variables.
EVITDC : Exponent vector inverse total degree compare.
EVL : Exponent Vector Length.
EVLCM : Exponent vector least common multiple.
EVLFCP : Exponent vector linear form compare.
EVLFCP : UGB exponent vector linear form compare.
EVLGIL : Exponent vector list generate for inverse lexicographical sequence.
EVLGTD : Exponent vector list generate for total degree.
EVLINV : Exponent vector list introduction of new variables.
EVLRNBSO : Rational exponent vector list bubble sort.
EVMT : Exponent vector multiple test.
EVMTJ : Exponent vector multiple test in the sense of Janet.
EVNNZE : Exponent vector number of non zero exponents.
EVOWRITE : Exponent vector order write.
EVPLM : Exponent vector pair-list merge.
EVPLSO : Exponent vector pair-list sort.
EVR : Exponent Vector Reduction.
EVRAND : Exponent vector random.
EVRASP : Exponent vector random.
EVRNC : Rational exponent vector compare.
EVRNGL : Rational exponent vector inverse graded lexicographical compare.
EVRWTDEG : Exponent vector rational-weighted total degree.
EVSIGN : Exponent vector signum.
EVSU : Exponent vector substitution.
EVSUM : Exponent vector sum.
EVT : Exponent Vector Test.
EVTDEG : Exponent vector total degree.
EVTSZ : Exponent vector test if starting with i zero exponents.
EVUMSORT : Exponent vector unique merge sort.
EVZERO : Exponent vector zero.
EXPTU : UGB extract exponent vector list from polynomial list.
EXVHOM : Exterior vector homomorphism.
GBE : Groebner Base with Exponent Vector Check.
GBEF : Groebner Base with Exponent Vector Check and Factors.
GENPOSV : Generate Postion Vector.
IKM : Integer vector component product.
INSPOSV : Insert Position Vector.
IVFRNV : Integer vector from rational number vector.
IVFRNV : Integer vector from rational number vector.
IVFRNV1 : Integer vector from rational number vector.
IVFRNV1 : Integer vector from rational number vector.
IVHOM : Integer vector homomorphism.
IVLC : Integer vector linear combination.
IVMAX : Integer vector maximum norm.
IVRAND : Integer vector random.
IVSPROD : Integer vector scalar product.
IVSQ : Integer vector scalar quotient.

IVSSUM : Integer vector scalar sum.
IVSVPROD : Integer vector scalar and vector product.
IVSVPROD : Integer vector scalar and vector product.
IVSVSUM : Integer vector scalar and vector sum.
IVSVSUM : Integer vector scalar and vector sum.
IVVDIF : Integer vector difference.
IVVPROD : Integer vector vectors product.
IVVSUM : Integer vector vector sum.
IVVSUM : Integer vector vector sum.
IVWRITE : Integer vector write.
MAKERN : UGB rational exponent vector list from integer ev list.
MDVHOM : Modular vector homomorphism.
MKSET : UGB rational exponent vector list difference list.
NEWDIF : UGB exponent vector list difference from polynomials.
NLGBE : Non-Commutative Groebner Base with Exponent Vector Check.
NLGBEF : Non-Commutative Groebner Base with Exponent Vector Check and Factors.
NULRNV : Rational number vector null test.
PDEGV : Polynomial degree vector.
PDIF : UGB rational exponent vector list difference list, incremental.
POWSEV : Power of variable symmetric product with exterior vector.
PVDEMA : Permutation vector for degree matrix.
PVFLISTS : permutation vector from lists.
RNSVPROD : Rational number vector product with scalar.
RNVDIF : Rational number vector difference.
RNVDIF : UGB rational exponent vector difference.
RNVFIV : Rational number vector from integer vector.
RNVFIV : Rational number vector from integer vector.
RNVLC : Rational number vector linear combination.
RNVMAX : Rational number vector maximum norm.
RNVQ : Rational number vector quotient.
RNVQF : Rational number vector quotient.
RNVREAD : Rational number vector read.
RNVSPROD : Rational number vector scalar product.
RNVSSUM : Rational number vector scalar sum.
RNVSVPROD : Rational number vector scalar vector product.
RNVSVPROD : Rational number vector scalar vector product.
RNVSVSUM : Rational number vector scalar sum.
RNVVPROD : Rational number vector vector product.
RNVVPROD : Rational number vector vector product.
RNVVSUM : Rational number vector vector sum.
RNVVSUM : Rational number vector vector sum.
RNVWRITE : Rational number vector write.
RRUADPOLTOVEC : Real root univariate arbitrary domain polynomial to vector.
RRUIPOLTOVEC : Real root univariate integral polynomial to vector.
RRVTEXT : Real root vector of tupels extension.
SCMULT : UGB rational exponent vector rational number product.
SCPROD : UGB rational exponent vector scalar product.
SKPRO2 : UGB rational exponent vector scalar product with integer ev.
SYGBE : Syzygy for Groebner Base with Exponent Vector.
UIPSIV : Univariate integral polynomial symmetric product with exterior integral vector.
VCOMP : Vector comparison.

VDELEL : Vector delete element.
VEL : Vector elements.
VIAZ : Vector of integers, adjoin zeros.
VIDIF : Vector of integers difference.
VIERED : Vector of integers, element reduction.
VILCOM : Vector of integers linear combination.
VINEG : Vector of integers negation.
VIPIIP : Vector of integral polynomials with vector of integers inner product.
VIPIIP : Vector of integral polynomials with vector of integers inner product.
VISPR : Vector of integers scalar product.
VISUM : Vector of integers sum.
VIUT : Vector of integers, unimodular transformation.
VMAX : Vector maximum.
VMIN : Vector maximum.
VMPIP : Vector of modular polynomial inner product.
VVECC : variable vector complement.
VVECFVLIST : variable vector from variable lists.

vectors

DEGRE : UGB total degree of a list of rational exponent vectors.
IVVPROD : Integer vector vectors product.

vektor

EVSSPROD : Exponent vektor set sorted product.

vel

ContractEt : contract vel.
ContractVel : contract vel.

verify

VERIFY : Verify conditions and polynomials.
VRNORM : Verify normalforms.

version

TfClassifyI : type formula classify coefficient tuple interpreter version.
TFGENI : TFGEN interpreter version.
TfZeroesI : TfZeroes interpreter version.

vertical

MTPLV : Matrix to Polynomial List Vertical.
PLVTM : Polynomial List Vertical To Matrix.
VMADD : Vertical Matrix Addition.

week

ILWCMP : Index list week compare.

weight

GSYTWG : G-Symmetric Term Weight.
SetDIPAGBOptions : Set the trace flag, the strategy and the variable weight list of the
SetDIPAGBVariableWeight : Set the DIPAGB variable weight list for the normal with sugar strategy.
SetUpdateVariableWeight : Set the DIPAGB variable weight update procedure.
UpdateVariableWeight : Update of the DIPAGB variable weight list.
WriteDIPAGBOptions : Write the current trace flag, strategy and variable weight list of the
WriteDIPAGBVariableWeight : Write the DIPAGB variable weight list in the output stream.

white

ColWhite : Colouring white.
FINDCP : Find white factors.
WHSRT : White sort.
WMEMB : White term member.
WUPD : White part update.

with

ADFACTO : Arbitrary domain factorization with variable order optimization.
ADPSUGNF : Arbitrary domain polynomial normal with sugar strategy normalform.
ADPSUGSP : Arbitrary domain polynomial normal with sugar strategy S-polynomial.
CP2 : UGB linear form product with rational exponent vector list 2.
CQ2 : UGB linear form product with rational exponent vector list.
DIFPF : Distributive polynomial with arbitrary domain coefficients from
DIIPWV : Distributive integral polynomial write with standard variable list.
DILFPFL : Distributive polynomial list with arbitrary domain coefficients from
DIPRLF : distributive polynomials reduce list of polynomials with factor.
DIRPMV : Distributiv Polynomial multiplication with a variable.
DIRPWV : Distributive rational polynomial write with standard variable
EDIIFSUGNF : Extended distributive integral function polynomial normal with sugar
EDIIFSUGSP : Extended distributive integral function polynomial normal with sugar
EDIPSUGCON : Extended distributive polynomial normal with sugar strategy constructor.
EDIPSUGNOR : Extended distributive polynomial normal with sugar strategy normal form.
EDIPSUGSP : Extended distributive polynomial normal with sugar strategy S-polynomial.
EDIT : Edit file with name s.
EVTSZ : Exponent vector test if starting with i zero exponents.
GBE : Groebner Base with Exponent Vector Check.
GBEF : Groebner Base with Exponent Vector Check and Factors.
GBF : Groebner Base with Factors.
GBSYSF : Groebner system with factorization.
GSYSF : Groebner system with factorization.
MCPMV : Matrix of coefficients of polynomials, with respect to main variable.
NLGBE : Non-Commutative Groebner Base with Exponent Vector Check.
NLGBEF : Non-Commutative Groebner Base with Exponent Vector Check and Factors.
NLGBF : Non-Commutative Groebner Base with Factors.
NLRCSPR : Reduction Chain of S-Polynomials with Remainder.

NLSPC : Non-Commutative S-Polynomial with Coefficients.
NLSPCEGB : Non-Commutative S-Polynomials with Coefficients and Exponenetvector-Check
NLSPCGB : Non-Commutative S-Polynomials with Coefficients for Groebner Base.
PLF : UGB linear form with precomputed linear forms.
POWSEV : Power of variable symmetric product with exterior vector.
PUGB : Universal Groebner base with precomputed linear forms.
RCSPR : Reduction Chain of S-Polynomials with Remainder.
RNSVPROD : Rational number vector product with scalar.
RQEPRRC : Real Quantifier Elimination with Parametric Real Root Count.
SetDIPAGBVariableWeight : Set the DIPAGB variable weight list for the normal with sugar strategy.
SetFactoFunc : Set factorization with variable order optimization function in domain.
SetPSugNormFunc : Set polynomial normal with sugar strategy normalform function in domain.
SetPSugSpolFunc : Set polynomial normal with sugar strategy S-polynomial function in domain.
SKPRO2 : UGB rational exponent vector scalar product with integer ev.
SPC : S-Polynomial with Coefficients.
SPCEGB : S-Polynomials with Coefficients and Exponenetvector-Check for Groebner Base.
SPCGB : S-Polynomials with Coefficients for Groebner Base.
SUBLIS : Substitution with list.
SUBLIS : Substitution with list.
SYGBE : Syzygy for Groebner Base with Exponent Vector.
TfCtj : type formula coefficient tuples with joker argument.
TFFTUPLE : type formula from coefficient tuple with joker entries.
TFGENJ : type formula generate with joker argument.
UIPSIL : Univariate integral polynomial symmetric product with exterior index list.
UIPSIV : Univariate integral polynomial symmetric product with exterior integral vector.
VIPIIP : Vector of integral polynomials with vector of integers inner product.

without

CGBCOL : Write coloured polynomials without green monomials.
GGREEN : Write groebner-system without green monomials.
GREPOL : Get polynomials without green monomials.
MKPOL : Make polynomial without green monomials.
NONEWL : UGB update linear forms without new terms.

write

ADDDWRIT : Arbitrary domain, domain descriptor write.
ADMWRITE : Arbitrary domain matrix write.
ADVWRITE : Arbitrary domain vector write.
ADWRIT : Arbitrary domain write.
ANDWR : Algebraic number decimal write.
APDWR : Algebraic point, decimal write.
APWRIT : Arbitrary precision floating point write.
AWRITE : Atom write.
CdpWrite : Case distinction and polynomial set write.
CdWrite : Case distinction write.

CDWRITE : Complex number decimal write.
CGBCOL : Write coloured polynomials without green monomials.
CGBOPTWRITE : Comprehensive Groebner Basis Options Write
CgbWrite : Comprehensive Groebner basis write.
CNWRITE : Complex number write.
CondWrite : Condition write.
CounterWR : Counter Write.
CWRIT2 : Character write, 2 characters.
CWRIT3 : Character write, 3 characters.
CWRIT4 : Character write, 4 characters.
CWRIT5 : Character write, 5 characters.
CWRIT6 : Character write, 6 characters.
CWRITE : Character write.
DCLWR : Coloured polynomials list write.
DIDIMWR : Distributive polynomial dimension write.
DIILWR : Distributive integral polynomial list write.
DIIPWR : Distributive integral polynomial write.
DIIPWV : Distributive integral polynomial write with standard variable list.
DIITWR : Distributive polynomial system interval tupels write.
DILWR : Distributive polynomial list write.
DINTWR : Distributive polynomial system normalized tupels write.
DIRLPW : DIP rational polynomial ideal primary ideal decomposition write.
DIRLWR : Distributive rational polynomial list write.
DIROWR : Distributive polynomial system real root write.
DIRPWR : Distributive rational polynomial write.
DIRPWV : Distributive rational polynomial write with standard variable
DIWRIT : Distributive polynomial write.
DoWrite : Do Write.
DWRIT : Distinction write.
ECPWRITE : Extended critical pair write.
EDIPWRITE : Extended distributive polynomial write.
EIMWRT : Exterior integral matrix write.
EIVWRT : Exterior integral vector write.
EvordWrite : Evord Write.
EVOWRITE : Exponent vector order write.
FdWrite : Formula and dimension write.
FFWRITE : Finite field write.
FILWRITE : Formatted indented list write.
FLWRITE : Formatted list write.
FORTEXW : formula tex write.
FORTEXWLVAR : tex write list of varaiables.
FORTEXWVAR : formula tex write variable.
GGREEN : Write groebner-system without green monomials.
GsWrite : Groebner system write.
GSYPGW : G-Symmetric Permutation Group Write.
GWRITE : Gamma-integer write.
IBLWR : Involutive bases list write.
IFWRIT : Integral function write.
ILWRIT : Integer list write.
IMWRITE : Integer matrix write.
INLWRT : Index list write.
IPWRIT : Integral polynomial write.
IVWRITE : Integer vector write.

IWRITE : Integer write.
LECPWRITE : List of extended critical pairs write.
LEDIPWRITE : List of extended distributive polynomials write.
LRNWRIT : List of rational numbers write.
LWRITE : List write.
MLDTEXW : maslog demonstration tex write.
NFWRIT : Normalforms write.
NWRIT0 : Normalforms write.
NWRIT1 : Normalforms write.
ODWRITE : Octonion number decimal write.
ONWRITE : Octonion number write.
OWRITE : Object write.
PdWrite : Parametric dimension write.
PFWRITE : Integral polynomial write.
PLWR : Polynomial List Write.
PMWR : Polynomial Matrix Write.
PQOPTWR : polynomial options write.
PQPRINGWR : polynomial equation polynomial ring write.
PQTEXW : polynomial equation tex write.
pqtexwaf : polynomia equation tex write atomic formula.
QDWRITE : Quaternion number decimal write.
QNWRITE : Quaternion number write.
RFWRIT : Rational function write.
RIRWRT : Rational interval refinement write.
RNDWR : Rational number decimal write.
RNDWR : Rational number decimal write.
RNDWRS : Rational number decimal write special.
RNMWRITE : Rational number matrix write.
RNVWRITE : Rational number vector write.
RNWRIT : Rational number write.
RPLWRS : Rational polynomial list write.
RPWRIT : Rational polynomial write.
RPWRTS : Rational polynomial write.
RQEOPTWR : real quantifier elimination option write.
SetDdwrFunc : Set domain descriptor write function in domain.
SetECPWrite : Set the extended critical pair write procedure.
SetEDIPWrite : Set the extended distributive polynomial write procedure.
SetWritFunc : Set write function in domain.
STWRT : Symbol tree write.
STWRT : Symbol tree write.
SUBSGW : Substitution Group Write.
SWRITE : String write.
SysInfoWrite : system information write.
SYWRIT : Symbol write.
SYWRIT : Symbol write.
UWRIT1 : Universal write, 1.
UWRIT1 : Universal write, 1.
UWRIT1 : Universal write, 1.
UWRITE : Universal write.
UWRITE : Universal write.
UWRITE : Universal write.
VLWRIT : Variable list write.
WPAIRS : Write pairs of polynomials.

WPLIST : Write polynomials and pairs.
WRCGB : Write comprehensive-groebner-basis.
WRCOL : Write colour.
WRCONJ : Write conjunction.
WRDIMS : Write dimension.
WRGBS : Write groebner-system.
WriteDCGBopt : write decompositional groebner bases options.
WriteDIPAGBOptions : Write the current trace flag, strategy and variable weight list of the
WriteDIPAGBStrategy : Write the DIPAGB strategy option for the extended critical pair selection
WriteDIPAGBTraceFlag: Write the DIPAGB trace flag in the output stream.
WriteDIPAGBVariableWeight: Write the DIPAGB variable weight list in the output stream.
WRQFN0 : Write quantifier free formula for parametric ideal membership.
WRRCGB : Write reduced comprehensive-groebner-basis.
WRS1 : Write Situation.
WRS2 : Write Situation.
WRTERM : Write term.
WRTEST : Write groebner test.
WRTITL : Write title.
WRUGB : Write universal Groebner base.
WRUGF : Write universal Groebner family.

zero

ColpIsZero : Coloured polynomial is zero.
CondZero : Condition zero part.
DIGBZT : Distributive polynomial groebner base common zero test.
DINTZS : DIP nomalized tupels from system zero.
DIPZ : DIP Zero Dimensional Ideal Definition Module.
DIRGZS : Distributive rational Groebner base zero set.
DITFZS : DIP tupel from zero set.
DITSPL : DIP zero set tupel split.
EVNNZE : Exponent vector number of non zero exponents.
EVTSZ : Exponent vector test if starting with i zero exponents.
EVZERO : Exponent vector zero.
GBZSET : Groebner base real zero set of zero dimensional ideal.
GBZSET : Groebner base real zero set of zero dimensional ideal.

zero-dimensional

RRZDIM : Real root zero-dimensional test.

zeroes

TfZeroes : type formula zeroes.
TfZeroes0 : type formula zeroes 0.

zeros

VIAZ : Vector of integers, adjoin zeros.

Index