```
--> load ( gcdex ) $ load ( grobner ) $

--> /*==========       subroutine 1 ==========================*/

--> Inverse ( A , Const ) : = block (

      [ Nconst , vn , Tn , rwn , rpn ] ,

    Nconst : length ( Const ) ,
    vn : Nconst ,
    varL : [ ] ,
    maxpL : [ ] ,

    Tn : 1 ,
    for i : 1 thru Nconst do (

      con : Const [ i ] ,
      y [ i ] : con [ 1 ] , varL : endcons ( y [ i ] , varL ) ,
      D [ i ] : con [ 2 ] ,
      pw [ i ] : hipow ( D [ i ] , y [ i ] ) , maxpL : endcons ( pw [ i ] , maxpL ) , Tn : Tn * pw [ i ]

      ) ,
      /*    基底の次数の組み合わせ



    Tn:    pw[1]*pw[2]*pw[3]   2*3*2=12
    pwL:   基底次数の組み合わせのリスト
           [[0,0,0],[0,0,1],[0,1,0],[0,1,1],[0,2,0],[0,2,1],
             [1,0,0],[1,0,1],[1,1,0],[1,1,1],[1,2,0],[1,2,1]]

    rwn:   繰り返し同じ数を書く数(re-wright number)
    rpn:   (repetition number)
           例えば上の例ではリスト[x1,x2,x3]のx3の例でいうとx2だけをみると
                  [0,0,1,1,2,2,0,0,1,1,2,2]となっているが、この
           [0,0,1,1,2,2]の繰り返しが2回繰り返されている。
           この数を指定する数
        Tn    pw[i]   rwn  rpr
        12    ÷ 2    =  6    1=Tn/(pw[1]*rwn)
         6    ÷ 3    =  2    2=Tn/(pw[2]*rwn)
         2    ÷ 2    =  1    6=Tn/(pw[3]*rwn)

        という関係式があるので下記のプログラムとなる
```

```
   */

pwL : [ ] ,
for i : 1 thru Tn do ( pwL : endcons ( [ ] , pwL ) ) ,

rwn : Tn ,
for i : 1 thru Nconst do (

  rwn : rwn / pw [ i ] ,
  rpn : Tn / ( pw [ i ] * rwn ) ,
  n : 1 ,

for j : 1 thru rpn do (

for k : 1 thru pw [ i ] do (

for m : 1 thru rwn do (
        pwL [ n ] : endcons ( k - 1 , pwL [ n ] ) , n : n + 1
        )
     )
     )
     ) ,
baseL : [ ] ,

for i : 1 thru Tn do (
  b : 1 ,
  for j : 1 thru vn do ( b : b * varL [ j ] ^ pwL [ i ] [ j ] ) ,
  baseL : endcons ( b , baseL )

) ,
coefL : makelist ( c [ i ] , i , 0 , Tn - 1 ) ,
F : sum ( coefL [ i ] * baseL [ i ] , i , 1 , Tn ) ,
W : F * A ,

for i : 1 thru Nconst do (
W : remainder ( W , Const [ i ] [ 2 ] , varL [ i ] )
) ,
W : expand ( W ) ,
CeL : [ ] ,
for i : 1 thru Tn do (
  ce : W ,
  for j : 1 thru vn do ( ce : coeff ( ce , varL [ j ] , pwL [ i ] [ j ] ) ) ,
```

```
    CeL : endcons ( ce , CeL )

    ) ,
    CeL [ 1 ] : CeL [ 1 ] - 1 ,
  SLV : solve ( CeL , coefL ) ,
  IA : sum ( coefL [ i ] * baseL [ i ] , i , 1 , Tn ) ,
  IA : ev ( IA , SLV )
  ) $
--> /*==========      subroutine 2 ==========================*/
--> Reduce ( A , Const ) : = block (

  crn : length ( Const ) ,

  for i : 1 thru crn do ( A : remainder ( A , Const [ i ] [ 2 ] , Const [ i ] [ 1 ] ) ) ,

  A

  ) $
--> /*==========      main program ==========================*/
--> F : x ^ 5 - 1 $ F : factor ( F ) ;
  F1 : part ( F , 1 ) ;  fx : part ( F , 2 ) ; gx [ 0 ] : fx $ gv [ 0 ] : subst ( v , x , fx ) ;
--> Const : [ ] $
  CurConst : cons ( [ v , gv [ 0 ] ] , Const ) ;
--> N : 4 $
--> vL : makelist ( v [ i ] , i , 1 , N ) ;
--> yL : makelist ( v ^ i , i , 1 , N ) ;
  zL : [ ] $
  for i : 1 thru N do (
  w : yL [ i ] , w : Reduce ( w , CurConst ) , zL : endcons ( w , zL )
  ) $
  zL ;
  vzL : map ( lambda ( [ x , y ] , x = y ) , vL , zL ) ;
  xvL : makelist ( x - v [ i ] , i , 1 , N ) ;
--> M : 5 $
  T : [ ] $

  for i : 1 thru ( M - 1 ) do (

  z : [ ] , n : 1 ,
```

```
    for j : 1 thru ( M - 1 ) do (

    n : n * i , n : mod ( n , M ) , z : endcons ( n , z )

    ) ,

  T : endcons ( z , T )

  ) $

  ST : [ ] $
  for i : 1 thru ( M - 1 ) do (
  print ( i , " : " , T [ i ] ) , s : sort ( T [ i ] ) , s : unique ( s ) , ST : endcons ( s , ST )
  ) $

  for i : 1 thru ( M - 1 ) do (
  print ( i , " : " , ST [ i ] )
  ) $

--> gr4 : [ 1 , 2 , 3 , 4 ] $
  gr2 : [ 1 , 4 ] $

--> gr2a : gr2 $
  gr2b : gr4 $
  for i : 1 thru 2 do (
  gr2b : delete ( gr2a [ i ] , gr2b )
  ) $
  gr2a ;
  gr2b ;

--> gr1a : [ 1 ] $ gr1b : [ 4 ] $

--> /* 1st step */

--> aL : [ ] $ bL : [ ] $
  for i : 1 thru 2 do (
  aL : endcons ( xvL [ gr2a [ i ] ] , aL ) , bL : endcons ( xvL [ gr2b [ i ] ] , bL )
  ) $
  h0 : apply ( "*" , aL ) ;
  h1 : apply ( "*" , bL ) ;

--> vzL ;

--> CurConst : cons ( [ v , gv [ 0 ] ] , Const ) ;
```

```
--> h0 : ev ( h0 , vzL ) ; h0 : Reduce ( h0 , CurConst ) ;
    h1 : ev ( h1 , vzL ) ; h1 : Reduce ( h1 , CurConst ) ;

--> t0 : ( h0 + h1 ) / 2 $ t1 : ( h0 - h1 ) / 2 $
    t0 : Reduce ( t0 , CurConst ) $ t1 : Reduce ( t1 , CurConst ) $
    t0 : expand ( t0 ) ; t1 : expand ( t1 ) ; t1 : factor ( t1 ) ;

--> t1 : expand ( t1 ) ;
    pmax : hipow ( t1 , x ) ;
    d : coeff ( t1 , x , pmax ) ;

--> Id : Inverse ( d , CurConst ) $ Id : ratsimp ( Id ) ;


    dId : d * Id $
    dId : Reduce ( dId , CurConst ) ;
    q : Id * t1 $
    A1 : d ^ 2 $ A1 : expand ( A1 ) ;
    q : Reduce ( q , CurConst ) ;
    A1 : Reduce ( A1 , CurConst ) ;

--> B [ 1 ] : expand ( a [ 1 ] ^ 2 - A1 ) ;

--> t1 : a [ 1 ] * q $ t1 : expand ( t1 ) ;

--> Const : cons ( [ a [ 1 ] , B [ 1 ] ] , Const ) ;

--> t0 : expand ( t0 ) ; t1 ;
    h0 : t0 + t1 $ h0 : expand ( h0 ) ;
    h1 : t0 - t1 $  h1 : expand ( h1 ) ;

--> gx [ 1 ] : h0 ; gx [ 1 ] : expand ( gx [ 1 ] ) ; gv [ 1 ] : subst ( v , x , gx [ 1 ] ) ;

--> /*  2nd step  */

--> Const ;
    CurConst : cons ( [ v , gv [ 1 ] ] , Const ) ;

--> aL : [ ] $ bL : [ ] $
    for i : 1 thru 1 do (
    aL : endcons ( xvL [ gr1a [ i ] ] , aL ) , bL : endcons ( xvL [ gr1b [ i ] ] , bL )
    ) $
    h0 : apply ( "*" , aL ) ;
    h1 : apply ( "*" , bL ) ;

--> h0 : ev ( h0 , vzL ) ; h1 : ev ( h1 , vzL ) ;
    h0 : Reduce ( h0 , CurConst ) $  h0 : expand ( h0 ) ;
    h1 : Reduce ( h1 , CurConst ) $ h1 : expand ( h1 ) ;

--> t0 : ( h0 + h1 ) / 2 $ t1 : ( h0 - h1 ) / 2 $
```

```
    t0 : Reduce ( t0 , CurConst ) $  t0 : expand ( t0 ) ;
    t1 : Reduce ( t1 , CurConst ) $ t1 : expand ( t1 ) ;

-->  t1 : expand ( t1 ) ;
    pmax : hipow ( t1 , x ) ;
    d : coeff ( t1 , x , pmax ) ;

-->  Id : Inverse ( d , CurConst ) $ Id : ratsimp ( Id ) ;
    dId : d * Id $
    dId : Reduce ( dId , CurConst ) ;
    q : Id * t1 $ A2 : d ^ 2 $
    A2 : expand ( A2 ) ;
    q : Reduce ( q , CurConst ) ;
    A2 : Reduce ( A2 , CurConst ) ;

-->  B [ 2 ] : expand ( a [ 2 ] ^ 2 - A2 ) ;

-->  t1 : a [ 2 ] * q $ t1 : expand ( t1 ) ;

-->  Const : cons ( [ a [ 2 ] , B [ 2 ] ] , Const ) ;

-->  t0 : expand ( t0 ) ; t1 ;
    h0 : t0 + t1 ;

-->  gx [ 2 ] : h0 ; gv [ 2 ] : subst ( v , x , gx [ 2 ] ) ;

-->  vsol : solve ( gv [ 2 ] , v ) ; vsol : expand ( vsol ) ;

-->  zL ;

-->  Const ;

-->  vzL ;

-->  ezL : ev ( zL , vsol ) $ ezL : expand ( ezL ) $
    rezL : [ ] $
    for i : 1 thru N do (
    w : ezL [ i ] , w : Reduce ( w , Const ) , rezL : endcons ( w , rezL )
    ) $ rezL ; rezL : expand ( rezL ) ;

-->  root : [ ] $
    a1r : allroots ( B [ 1 ] ) $ root : endcons ( a [ 1 ] = rhs ( a1r [ 1 ] ) , root ) ;
    b2 : ev ( B [ 2 ] , root ) $ b2r : allroots ( b2 ) $ root : endcons ( a [ 2 ] = rhs ( b2r [ 1 ] ) , root ) ;

-->  AnsL : ev ( rezL , root ) $ AnsL : expand ( AnsL ) ;

-->  allroots ( fx ) ;
```
