```
--> 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 ^ 17 - 1 $ F : factor ( F ) ;
/*
N:17$
T:[]$

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

z:[], n:1,

  for j:1 thru (N-1) do(

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

  ),

T:endcons(z,T)

)$

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

```
    )$

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

    */
--> N : 16 $

--> vL : makelist ( v [ i ] , i , 1 , N ) ;

--> /* zL:makelist(ζ^i,i,1,N); */

--> /* vzL:map(lambda([x,y],x=y),vL,zL); */

--> zL : makelist ( v ^ i , i , 1 , N ) ;
    vzL : map ( lambda ( [ x , y ] , x = y ) , vL , zL ) ;

--> xvL : makelist ( x - v [ i ] , i , 1 , N ) ;

--> g [ 0 ] : apply ( "*" , xvL ) ;

--  gx [ 0 ] : x ^ 16 + x ^ 15 + x ^ 14 + x ^ 13 + x ^ 12 + x ^ 11 + x ^ 10 + x ^ 9 + x ^ 8 + x ^ 7 + x ^ 6 +
>   x ^ 5 + x ^ 4 + x ^ 3 + x ^ 2 + x + 1 ;
    gv [ 0 ] : subst ( v , x , gx [ 0 ] ) ;

--> gxv : factor ( gx [ 0 ] , gv [ 0 ] ) ;

--  v [ 16 ] : - ( v ^ 15 + v ^ 14 + v ^ 13 + v ^ 12 + v ^ 11 + v ^ 10 + v ^ 9 + v ^ 8 + v ^ 7 + v ^ 6 + v ^ 5
>   + v ^ 4 + v ^ 3 + v ^ 2 + v + 1 ) $

--> vzL ;

--> gr16 : [ 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 , 12 , 13 , 14 , 15 , 16 ] $
    gr8 : [ 1 , 2 , 4 , 8 , 9 , 13 , 15 , 16 ] $
    gr4 : [ 1 , 4 , 13 , 16 ] $
    gr2 : [ 1 , 16 ] $

--> gr8a : gr8 $
    gr8b : gr16 $
    for i : 1 thru 8 do (
    gr8b : delete ( gr8a [ i ] , gr8b )
    ) $
    gr8a ;
    gr8b ;

--> gr4a : gr4 $
    gr4b : gr8 $
    for i : 1 thru 4 do (
```

```
    gr4b : delete ( gr4a [ i ] , gr4b )
    ) $
    gr4a ;
    gr4b ;

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

-->  gr1a : [ 1 ] $ gr1b : [ 16 ] $

-->  /* 1st step */



    aL : [ ] $ bL : [ ] $
    for i : 1 thru 8 do (
    aL : endcons ( xvL [ gr8a [ i ] ] , aL ) , bL : endcons ( xvL [ gr8b [ i ] ] , bL )
    ) $
    h0 : apply ( "*" , aL ) ;
    h1 : apply ( "*" , bL ) ;

-->  t0 : ( h0 + h1 ) / 2 $ t1 : ( h0 - h1 ) / 2 $
    t0 : expand ( ev ( t0 , vzL ) ) $ t1 : expand ( ev ( t1 , vzL ) ) $

    Const : [ ] $
    CurConst : cons ( [ v , gv [ 0 ] ] , Const ) ;
    t0 : Reduce ( t0 , CurConst ) ;
    t1 : Reduce ( t1 , CurConst ) ;

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

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

-->  B [ 1 ] : a [ 1 ] ^ 2 - A1 ;
```

```
--> cn : [ a [ 1 ] ] $ cn : endcons ( B [ 1 ] , cn ) $ Const : cons ( cn , Const ) ;

--> expand ( t0 ) ;
    t1 : a [ 1 ] * q ;
    h0 : expand ( t0 + t1 ) ;

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

--> /* 2nd step */


    aL : [ ] $ bL : [ ] $
    for i : 1 thru 4 do (
    aL : endcons ( xvL [ gr4a [ i ] ] , aL ) , bL : endcons ( xvL [ gr4b [ i ] ] , bL )
    ) $
    h0 : apply ( "*" , aL ) ;
    h1 : apply ( "*" , bL ) ;

--> t0 : ( h0 + h1 ) / 2 $ t1 : ( h0 - h1 ) / 2 $
    t0 : expand ( ev ( t0 , vzL ) ) $ t1 : expand ( ev ( t1 , vzL ) ) $

    CurConst : cons ( [ v , gv [ 1 ] ] , Const ) ;
    t0 : Reduce ( t0 , CurConst ) ;
    t1 : Reduce ( t1 , CurConst ) ;

--> 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 $
    q : Reduce ( q , CurConst ) ;
    A2 : d ^ 2 $
    A2 : Reduce ( A2 , CurConst ) ;

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

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

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

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

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

--> /*  3rd 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 ) ;

--> t0 : ( h0 + h1 ) / 2 $ t1 : ( h0 - h1 ) / 2 $
    t0 : expand ( ev ( t0 , vzL ) ) $ t1 : expand ( ev ( t1 , vzL ) ) $

    CurConst : cons ( [ v , gv [ 2 ] ] , Const ) ;
    t0 : Reduce ( t0 , CurConst ) ;
    t1 : Reduce ( t1 , CurConst ) ;

--> 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 $ A3 : d ^ 2 $
    q : Reduce ( q , CurConst ) ;
    A3 : Reduce ( A3 , CurConst ) ;

--> B [ 3 ] : expand ( a [ 3 ] ^ 2 - A3 ) ;

--> t1 : a [ 3 ] * q $ t1 : expand ( t1 ) ;

--> cn : [ a [ 3 ] ] $ cn : endcons ( B [ 3 ] , cn ) $  Const : cons ( cn , Const ) ;

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

--> gx [ 3 ] : h0 ; gv [ 3 ] : subst ( v , x , gx [ 3 ] ) ;

--> /*  4th step  */

--> 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 ) ;

--> t0 : ( h0 + h1 ) / 2 $ t1 : ( h0 - h1 ) / 2 $
    t0 : expand ( ev ( t0 , vzL ) ) $ t1 : expand ( ev ( t1 , vzL ) ) $
```

```
    CurConst : cons ( [ v , gv [ 3 ] ] , Const ) ;
    t0 : Reduce ( t0 , CurConst ) ;
    t1 : Reduce ( t1 , CurConst ) ;

-->  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 $ A4 : d ^ 2 $
    q : Reduce ( q , CurConst ) ;
    A4 : Reduce ( A4 , CurConst ) ;

-->  B [ 4 ] : expand ( a [ 4 ] ^ 2 - A4 ) ;

-->  t1 : a [ 4 ] * q $ t1 : expand ( t1 ) ;

-->  cn : [ a [ 4 ] ] $ cn : endcons ( B [ 4 ] , cn ) $  Const : cons ( cn , Const ) ;

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

-->  gx [ 4 ] : h0 ; gv [ 4 ] : subst ( v , x , gx [ 4 ] ) ;

-->  vsol : solve ( gv [ 4 ] , v ) ;

-->  zL ;

-->  Const ;

-->  zANS : [ ] $
    za : rhs ( vsol [ 1 ] ) $

    for i : 1 thru 16 do (

      zc : za ^ i , zc : Reduce ( zc , Const ) ,
     zc : expand ( zc ) ,
    zANS : endcons ( zc , zANS )

    ) $

    for i : 1 thru 16 do ( print ( i , "=" , zANS [ i ] ) ) $
```