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

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

--> /*
    以下は逆元を求めるサブルーチンプログラム
    使い方

    A:    逆元を求めたい元
    Const: 代数体を規定しているリスト

    例えば   Const:[ [v,gv[0]],[a[1],B[1]]] など
    gv[0]: 最小多項式   変数v
    B[1]:  添加される数a[1]が満たす冪根方程式

    具体的には   Const:[[v,v^8-v^6+v^4-v^2+1],[a[1],a[1]^2-5/4]]$

    */
--> 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 ==========================*/
--> /*
    以下は、代数体の次数を低減させるサブルーチンプログラム

    Const: 代数体を規定しているリスト

    例えば　Const:[ [v,gv],[a[1],B[1]]] など
    gv[i]: 現在の体で使用されている v　の最小多項式
    B[1]: 添加される数a[1]が満たす冪根方程式

    具体的には　Const:[[v,v^8-v^6+v^4-v^2+1],[a[1],a[1]^2-5/4]]$


      */
--> 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 ==========================*/
--> fx : x ^ 3 - 3 * x + 1 ;
    fv : subst ( x + v , x , fx ) ;
--> fx1 : divide ( fx , x - α , x ) ;
    fx2 : divide ( fx1 [ 1 ] , x - β , x ) ;
    fx3 : divide ( fx2 [ 1 ] , x - γ , x ) ;
--> gx : x ^ 3 - 9 * x - 9 ;
    gv : subst ( v , x , gx ) ;
--> fcx : factor ( fx , gv ) ; solve ( fcx , x ) ;
--> Const : [ ] $
    CurConst : cons ( [ v , gv ] , Const ) ;
--> R : resultant ( fv , gv , v ) ;
--> cp : coeff ( R , x , hipow ( R , x ) ) ;
    R : factor ( R ) ;
    if cp < 0 then R : R * ( - 1 ) ;
    R [ 1 ] : part ( R , 1 ) ; R [ 2 ] : part ( R , 2 ) ; R [ 3 ] : part ( R , 3 ) ;
--  /*  ------------------------------------ */
>   r [ 0 ] : R [ 1 ] ;
    r [ 1 ] : fv ; r [ 1 ] : expand ( r [ 1 ] ) ; r [ 1 ] : remainder ( r [ 1 ] , gv , v ) $ r [ 1 ] : expand ( r [ 1 ] )
    ;
--  /* 1st step */
>
    Dv : divide ( r [ 0 ] , r [ 1 ] , x ) $
    Dv [ 1 ] : Reduce ( Dv [ 1 ] , CurConst ) $ Dv [ 1 ] : expand ( Dv [ 1 ] ) $
    Dv [ 2 ] : Reduce ( Dv [ 2 ] , CurConst ) $ Dv [ 2 ] : expand ( Dv [ 2 ] ) $
    q [ 1 ] : Dv [ 1 ] ; pw : hipow ( Dv [ 2 ] , x ) $
    if Dv [ 2 ] = 0 then rc : 1 else rc : coeff ( Dv [ 2 ] , x , pw ) $ irc : Inverse ( rc , CurConst ) $
    r [ 2 ] : expand ( Dv [ 2 ] * irc ) $ r [ 2 ] : Reduce ( r [ 2 ] , CurConst ) $ r [ 2 ] : expand ( r [ 2 ] ) ;
    cc [ 2 ] : rc ; icc [ 2 ] : irc ;
--  /* 2nd step */
>   Dv : divide ( r [ 1 ] , r [ 2 ] , x ) $
    Dv [ 1 ] : Reduce ( Dv [ 1 ] , CurConst ) $ Dv [ 1 ] : expand ( Dv [ 1 ] ) $
    Dv [ 2 ] : Reduce ( Dv [ 2 ] , CurConst ) $ Dv [ 2 ] : expand ( Dv [ 2 ] ) $
    q [ 2 ] : Dv [ 1 ] ; Dv [ 2 ] : expand ( Dv [ 2 ] ) $ pw : hipow ( Dv [ 2 ] , x ) $
```

```
    if Dv[2]=0 then rc:1 else rc:coeff(Dv[2],x,pw)$ irc:Inverse(rc,CurConst)$
    r[3]:expand(Dv[2]*irc)$ r[3]:Reduce(r[3],CurConst)$ r[3]:expand(r[3]);
    cc[3]:rc; ic[3]:irc;

--  /* 3rd  step */
>   Dv:divide(r[2],r[3],x)$
    Dv[1]:Reduce(Dv[1],CurConst)$ Dv[1]:expand(Dv[1])$
    Dv[2]:Reduce(Dv[2],CurConst)$ Dv[2]:expand(Dv[2])$

    q[3]:Dv[1]; Dv[2]:expand(Dv[2])$ pw:hipow(Dv[2],x)$
    if Dv[2]=0 then rc:1 else rc:coeff(Dv[2],x,pw)$ irc:Inverse(rc,CurConst)$
    r[4]:expand(Dv[2]*irc)$ r[4]:Reduce(r[4],CurConst)$ r[4]:expand(r[4]);
    cc[4]:rc; ic[4]:irc;

--> z:solve(r[3],x); ans1:expand(rhs(z[1])+v);

--> /* ------------------------------------ */

--> /*  ------------------------------------ */
    r[0]:R[2];
    r[1]:fv; r[1]:expand(r[1]);

--  /* 1st step */
>
    Dv:divide(r[0],r[1],x)$
    Dv[1]:Reduce(Dv[1],CurConst)$ Dv[1]:expand(Dv[1])$
    Dv[2]:Reduce(Dv[2],CurConst)$ Dv[2]:expand(Dv[2])$
    q[1]:Dv[1]; pw:hipow(Dv[2],x)$
    if Dv[2]=0 then rc:1 else rc:coeff(Dv[2],x,pw)$ irc:Inverse(rc,CurConst)$
    r[2]:expand(Dv[2]*irc)$ r[2]:Reduce(r[2],CurConst)$ r[2]:expand(r[2]);
    cc[2]:rc; icc[2]:irc;

--  /* 2nd step */
>   Dv:divide(r[1],r[2],x)$
    Dv[1]:Reduce(Dv[1],CurConst)$ Dv[1]:expand(Dv[1])$
    Dv[2]:Reduce(Dv[2],CurConst)$ Dv[2]:expand(Dv[2])$
    q[2]:Dv[1]; Dv[2]:expand(Dv[2])$ pw:hipow(Dv[2],x)$
    if Dv[2]=0 then rc:1 else rc:coeff(Dv[2],x,pw)$ irc:Inverse(rc,CurConst)$
    r[3]:expand(Dv[2]*irc)$ r[3]:Reduce(r[3],CurConst)$ r[3]:expand(r[3]);
    cc[3]:rc; ic[3]:irc;

--  /* 3rd  step */
>   Dv:divide(r[2],r[3],x)$
    Dv[1]:Reduce(Dv[1],CurConst)$ Dv[1]:expand(Dv[1])$
    Dv[2]:Reduce(Dv[2],CurConst)$ Dv[2]:expand(Dv[2])$

    q[3]:Dv[1]; Dv[2]:expand(Dv[2])$ pw:hipow(Dv[2],x)$
```

```
    if Dv[2]=0 then rc:1 else rc:coeff(Dv[2],x,pw)$ irc:Inverse(rc,CurConst)$
    r[4]:expand(Dv[2]*irc)$ r[4]:Reduce(r[4],CurConst)$ r[4]:expand(r[4]);
    cc[4]:rc; ic[4]:irc;

--> z:solve(r[3],x); ans2:expand(rhs(z[1])+v);

--> /* ------------------------------------ */

--> /*  ------------------------------------ */
    r[0]:R[3];
    r[1]:fv; r[1]:expand(r[1]);

--  /* 1st step */
>
    Dv:divide(r[0],r[1],x)$
    Dv[1]:Reduce(Dv[1],CurConst)$ Dv[1]:expand(Dv[1])$
    Dv[2]:Reduce(Dv[2],CurConst)$ Dv[2]:expand(Dv[2])$
    q[1]:Dv[1]; pw:hipow(Dv[2],x)$
    if Dv[2]=0 then rc:1 else rc:coeff(Dv[2],x,pw)$ irc:Inverse(rc,CurConst)$
    r[2]:expand(Dv[2]*irc)$ r[2]:Reduce(r[2],CurConst)$ r[2]:expand(r[2]);
    cc[2]:rc; icc[2]:irc;

--  /* 2nd step */
>   Dv:divide(r[1],r[2],x)$
    Dv[1]:Reduce(Dv[1],CurConst)$ Dv[1]:expand(Dv[1])$
    Dv[2]:Reduce(Dv[2],CurConst)$ Dv[2]:expand(Dv[2])$
    q[2]:Dv[1]; Dv[2]:expand(Dv[2])$ pw:hipow(Dv[2],x)$
    if Dv[2]=0 then rc:1 else rc:coeff(Dv[2],x,pw)$ irc:Inverse(rc,CurConst)$
    r[3]:expand(Dv[2]*irc)$ r[3]:Reduce(r[3],CurConst)$ r[3]:expand(r[3]);
    cc[3]:rc; ic[3]:irc;

--  /* 3rd  step */
>   Dv:divide(r[2],r[3],x)$
    Dv[1]:Reduce(Dv[1],CurConst)$ Dv[1]:expand(Dv[1])$
    Dv[2]:Reduce(Dv[2],CurConst)$ Dv[2]:expand(Dv[2])$

    q[3]:Dv[1]; Dv[2]:expand(Dv[2])$ pw:hipow(Dv[2],x)$
    if Dv[2]=0 then rc:1 else rc:coeff(Dv[2],x,pw)$ irc:Inverse(rc,CurConst)$
    r[4]:expand(Dv[2]*irc)$ r[4]:Reduce(r[4],CurConst)$ r[4]:expand(r[4]);
    cc[4]:rc; ic[4]:irc;

--> z:solve(r[3],x); ans3:expand(rhs(z[1])+v);

--> /*ans1,ans2,ans3が最終的に　答え(factor(fx,gv))と一致しているか見てみる*/

--> ans1; ans2; ans3;

--> cx:factor(fx,gv)$ expand(solve(fcx,x));
```

```
--> /*2つを比較して同じ結果となったこれでOK*/

--> /* ------------------------------------ */

--> Q1 : matrix ( [ q1 , c2 ] , [ 1 , 0 ] ) ; iQ1 : matrix ( [ 0 , c2 ] , [ 1 , - q1 ] ) ;
    Q2 : matrix ( [ q2 , c3 ] , [ 1 , 0 ] ) ; iQ2 : matrix ( [ 0 , c3 ] , [ 1 , - q2 ] ) ;
    Q3 : matrix ( [ q3 , c4 ] , [ 1 , 0 ] ) ; iQ3 : matrix ( [ 0 , c4 ] , [ 1 , - q3 ] ) ;

--> Qm : Q1 . Q2 . Q3 ;
    iQm : iQ3 . iQ2 . iQ1 ; iQm : iQm / ( c2 * c3 * c4 ) ; iQm : expand ( iQm ) ;

--> T : Qm . iQm ; T : expand ( T ) ;

--> vr0 : [ r0 , r1 ] ; vr1 : [ r1 , r2 ] ; v2 : [ r2 , r3 ] ; vr3 : [ r3 , r4 ] ;

--> /* r4=0であるので、r4の段階で止めても問題なし　ver1で十分*/

--> ans : iQm . vr0 ;
```
