```
--> 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 )
    ) $
--> /*
    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]]$

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

--> fx : x ^ 5 + x ^ 4 + 2 * x ^ 3 + 4 * x ^ 2 + x + 1 $

--> N : hipow ( fx , x ) ; NN : N ! ;

--> /*  第1ステップ      */ ;

--> fx1 : divide ( fx , x - a , x ) ;
    fx2 : divide ( fx1 [ 1 ] , x - b , x ) ;
    fx3 : divide ( fx2 [ 1 ] , x - c , x ) ;
    fx4 : divide ( fx3 [ 1 ] , x - d , x ) ;
    fx5 : divide ( fx4 [ 1 ] , x - s , x ) ;

--> q1 : eliminate ( [ v - a - 2 * b - 3 * c - 4 * d - 5 * s , fx5 [ 2 ] ] , [ s ] ) $
    q2 : eliminate ( [ q1 [ 1 ] , fx4 [ 2 ] ] , [ d ] ) $
    q3 : eliminate ( [ q2 [ 1 ] , fx3 [ 2 ] ] , [ c ] ) $
    q4 : eliminate ( [ q3 [ 1 ] , fx2 [ 2 ] ] , [ b ] ) $
    q5 : eliminate ( [ q4 [ 1 ] , fx1 [ 2 ] ] , [ a ] ) $

--> Gv : q5 [ 1 ] $

--> Gv : factor ( Gv ) ;

--> pV : hipow ( Gv , V ) $
    if pV = NN then gv : Gv else gv : part ( Gv , 1 ) $
    gv ;

--> /*
    η[1]:ζ$  η[2]:ζ^2$  η[3]:ζ^3$  η[4]:ζ^4$
    Z:1+η[1]+η[2]+η[3]+η[4]$
    gvz:factor(gv,Z);
    */

--> fxg : factor ( fx , gv ) $
    sofxgL : solve ( fxg , x ) $

--> anL : [ ] $
    for i : 1 thru N do ( anL : endcons ( rhs ( sofxgL [ i ] ) , anL ) ) $
    anL $

--> /* makelist   の良い例
          map関数+lambda関数   の解かりやすい例*/

    kiL : makelist ( k [ i ] , i , 1 , N ) ;
    ksoL : map ( lambda ( [ x , y ] , x = y ) , kiL , anL ) $
```

```
--> pkiL : listify ( permutations ( kiL ) ) $
    Vdef ( w ) : = w [ 1 ] + 2 * w [ 2 ] + 3 * w [ 3 ] + 4 * w [ 4 ] + 5 * w [ 5 ] $

--> /*      apply関数の例      apply(f,[a,b,c])=f(a,b,c)になる          */
    pkiL [ 4 ] ;
    Vdef ( pkiL [ 4 ] ) ;

--> fix : 0 $
    for i : 1 thru 120 do ( vkev : expand ( ev ( Vdef ( pkiL [ i ] ) , ksoL ) ) ,
                        if vkev = v then fix : i
                        ) $
    fix ;  pkiL [ fix ] ;

--> rtL : [ α , β , γ , δ , ε ] $
    fixki : map ( lambda ( [ x , y ] , x = y ) , rtL , pkiL [ fix ] ) ;
    SoL : ev ( fixki , ksoL ) $

--> prtL : listify ( permutations ( rtL ) ) $

--> viL : makelist ( v [ i ] , i , 1 , NN ) ;

--> pVrtL : map ( lambda ( [ z ] , Vdef ( z ) ) , prtL ) $

--> virtL : map ( lambda ( [ x , y ] , x = y ) , viL , pVrtL ) $
    VivL : expand ( ev ( virtL , SoL ) ) $
    /*
    virtL[1]; virtL[10];
    VivL[1];  VivL[10];
    */

--> pvevL : [ ] $
    for i : 1 thru NN do ( pvev : expand ( ev ( pVrtL [ i ] , SoL ) ) ,
      vsub : subst ( pvev , v , gv ) , vsub : remainder ( vsub , gv , v ) ,
       pvevL : endcons ( vsub , pvevL )
      ) $

--
>   rtsqL : [ ] $ vmapL : [ ] $
    for i : 1 thru NN do (

      if pvevL [ i ] = 0 then ( rtsqL : endcons ( i , rtsqL ) , vmapL : endcons ( pvevL [ i ] , vmapL ) )
    ) $
    rtsqL ; vmapL ;

--> /*
    これで準備が整った。後は組成列に従って巡回拡大に分解して
    gv=gv[0] - > gv[1] -. gv[2] と
```

```
    最小多項式の次数を提言してゆく計算
    */
-->  /*  第1ステップ    P=2   */
-->  /*ここ重要   拘束条件のリスト作成*/
    gv[0]:gv$

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

    /*現時点での単拡大体F(v)上での拘束条件作成は以下の通り*/
-->  Gal11:[1,43,52,90,117,8,30,61,95,108]$
    Gal12:[18,33,70,80,99,23,40,59,73,110]$
-->  N1:length(Gal11);
-->  vx1L:makelist(x-v[Gal11[i]],i,1,N1);
    vx2L:makelist(x-v[Gal12[i]],i,1,N1);
-->  h0:apply("*",vx1L);
    h1:apply("*",vx2L);
-->  h0:ev(h0,VivL)$  h0:Reduce(h0,CurConst)$
    h1:ev(h1,VivL)$  h1:Reduce(h1,CurConst)$
-->  t0:ratexpand((h0+h1)/2)$ t0:Reduce(t0,CurConst)$
    t1:ratexpand((h0-h1)/2)$ t1:Reduce(t1,CurConst)$
-->  ratvars(x)$ t0;t1:ratsimp(t1)$
    /* t1:expand(t1)$ t1:ratsimp(t1); */
-->  t1:expand(t1)$
    cv:coeff(t1,x,hipow(t1,x))$
    cv2:cv^2$ cv2:Reduce(cv2,CurConst)$
    A[1]:cv2$ B[1]:a[1]^2-A[1];
-->  icv:Inverse(cv,CurConst)$
    q1:Reduce(icv*t1,CurConst)$ q1:expand(q1);
-->  t1:a[1]*q1;
    h0:t0+t1$ gx[1]:h0;gv[1]:subst(v,x,h0);
-->  /*
    新たな制約条件B[1]と新たな最小多項式gv[1]が生成されたので、
    Const  CurConst   を後進する必要があるので以下の命令をする
    */
```

```
    cn:[a[1],B[1]]$Const:cons(cn,Const);
    CurConst:cons([v,gv[1]],Const);

    /*拘束条件作成は終*/


-->  /*  第2ステップ    P=2   */

-->  Gal21:[1,43,52,90,117]$
     Gal22:[8,30,61,95,108]$

-->  N2:length(Gal21);

-->  vx1L:makelist(x-v[Gal21[i]],i,1,N2);
     vx2L:makelist(x-v[Gal22[i]],i,1,N2);

-->  h0:apply("*",vx1L);
     h1:apply("*",vx2L);

-->  h0:ev(h0,VivL)$  h0:Reduce(h0,CurConst)$
     h1:ev(h1,VivL)$  h1:Reduce(h1,CurConst)$

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

-->  ratvars(x)$ t0;t1:ratsimp(t1);
     /* t1:expand(t1)$ t1:ratsimp(t1); */

-->  t1:expand(t1)$
     cv:coeff(t1,x,hipow(t1,x));
     cv2:cv^2$ cv2:Reduce(cv2,CurConst);
     A[2]:cv2$ B[2]:a[2]^2-A[2];

-->  icv:Inverse(cv,CurConst);
     q1:Reduce(icv*t1,CurConst)$ q1:expand(q1);

-->  t1:a[2]*q1;
     h0:t0+t1$ gx[2]:h0; gv[2]:subst(v,x,h0);

-->  /*
     新たな制約条件B[1]と新たな最小多項式gv[1]が生成されたので、
     Const  CurConst    を後進する必要があるので以下の命令をする
     */

     cn:[a[2],B[2]]$Const:cons(cn,Const);
     CurConst:cons([v,gv[2]],Const);

     /*拘束条件作成は終*/
```

```
--> /*  第3ステップ    P=5   */

--> η[1]:ζ$η[2]:ζ^2$η[3]:ζ^3$η[4]:ζ^4$
    Z:1+η[1]+η[2]+η[3]+η[4]$
    cn:[ζ,Z]$Const:cons(cn,Const)$
    CurConst:cons([v,gv[2]],Const);

--> h0:x-v[1]$    h0:ev(h0,VivL)$    h0:Reduce(h0,CurConst)$
    h1:x-v[43]$    h1:ev(h1,VivL)$    h1:Reduce(h1,CurConst)$
    h2:x-v[117]$    h2:ev(h2,VivL)$    h2:Reduce(h2,CurConst)$
    h3:x-v[90]$    h3:ev(h3,VivL)$    h3:Reduce(h3,CurConst)$
    h4:x-v[52]$    h4:ev(h4,VivL)$    h4:Reduce(h4,CurConst)$

--  t0:(1/5)*(h0+h1+h2+h3+h4)$                    t0:Reduce(t0,CurConst
>   );
    t1:(1/5)*(h0+h1*η[1]+h2*η[1]^2+h3*η[1]^3+h4*η[1]^4)$   t1:
    Reduce(t1,CurConst)$
    t2:(1/5)*(h0+h1*η[2]+h2*η[2]^2+h3*η[2]^3+h4*η[2]^4)$   t2:
    Reduce(t2,CurConst)$
    t3:(1/5)*(h0+h1*η[3]+h2*η[3]^2+h3*η[3]^3+h4*η[3]^4)$   t3:
    Reduce(t3,CurConst)$
    t4:(1/5)*(h0+h1*η[4]+h2*η[4]^2+h3*η[4]^3+h4*η[4]^4)$   t4:
    Reduce(t4,CurConst);

--> hipow(t1,x);

--> Const;CurConst;

--> T1:t1^5$T1:Reduce(T1,CurConst);
    A[3]:T1$B[3]:a[3]^5-A[3];
    IA:Inverse(A[3],CurConst);  z:Reduce(A[3]*IA,CurConst);

--> T12:t1^3*t2$T12:Reduce(T12,CurConst);
    T13:t1^2*t3$T13:Reduce(T13,CurConst);
    T14:t1*t4$    T14:Reduce(T14,CurConst);

--> a2:a[3]^2*T12*IA$   a2:Reduce(a2,CurConst);
    a3:a[3]^3*T13*IA$   a3:Reduce(a3,CurConst);
    a4:a[3]^4*T14*IA$   a4:Reduce(a4,CurConst);

--> t1:a[3]$t2:a2$t3:a3$t4:a4$
    h0:t0+t1+t2+t3+t4;

--> gv[3]:subst(v,x,h0);
    cn:[a[3],B[3]]$Const:cons(cn,Const)$
    CurConst:cons([v,gv[3]],Const);

--> CurConst;
```

```
--> RVL : [ 1 ] $ rvz : 1 $
    for i : 1 thru 19 do (
    rvz : rvz * v ,   rvz : Reduce ( rvz , CurConst ) , RVL : endcons ( rvz , RVL )
    ) $

--> FansL : [ ] $
    for i : 1 thru N do (
    fan : anL [ i ] , fan : expand ( fan ) , sum : 0 ,
      for j : 1 thru 20 do (
    can : coeff ( fan , v , ( j - 1 ) ) , sum : can * RVL [ j ] + sum

        ) , sum : Reduce ( sum , CurConst ) , FansL : endcons ( sum , FansL )
    ) $

--> Num : [ ] $
    wf : allroots ( Z ) ; Num : endcons ( wf [ 1 ] , Num ) ;

--> B1f : ev ( B [ 1 ] , Num ) ; b1f : allroots ( B1f ) ; Num : endcons ( b1f [ 1 ] , Num ) ;

--> B2f : ev ( B [ 2 ] , Num ) $ b2f : allroots ( B2f ) $ Num : endcons ( b2f [ 1 ] , Num ) ;

--  B3f : ev ( B [ 3 ] , Num ) $ B3f : expand ( B3f ) $ b3f : allroots ( B3f ) $ Num : endcons ( b3f [ 1
>   ] , Num ) ;

--> FansL : ev ( FansL , Num ) $   FansL : expand ( FansL ) ;

--> allroots ( fx ) ;
```
