```
--> load ( gcdex ) $
--> /*==========       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 ;
    v0 : α + 2 * β + 3 * γ ;
    N : 3 $ NN : N ! $

--> /*
    3次方程式 fx=0の根を ( α, β, γ  ) とするとき
    (α,β,γ)の全ての置換の組をリスト化する命令

    permutations([α,β,γ])

    */

--> abcL : [ α , β , γ ] $
    abcpL : listify ( permutations ( [ α , β , γ ] ) ) ;

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

--> V ( u , w , y ) : = u + 2 * w + 3 * y $

--> VpL : map ( lambda ( [ x ] , apply ( V , x ) ) , abcpL ) ;

--> VabcL : map ( lambda ( [ x , y ] , x = y ) , viL , VpL ) ;

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

--> /*  第1ステップ

    fxを(x-α), (x-β), (x-γ)で順番に割り算してゆく。
    divideという命令を使う

    その余りが全てゼロと言う条件を課すと良い。

    fx/(x-a)の余り                          α^3-3*α-1
    その商を(x-b)で割ったときの余り          β^2+α*β+α^2-3
    更にその商を(x-c)で割ったときの余り      γ+β+α

    を使う */

--> fx1 : divide ( fx , x - α , x ) ;

--> fx2 : divide ( fx1 [ 1 ] , x - β , x ) ;

--> fx3 : divide ( fx2 [ 1 ] , x - γ , x ) ;
```

```
--> /*第2ステップ　其の二
    終結式を使うのだが　eliminateでなくて
    resultantを使った方が使いやすい
    */

--> r1 : resultant ( v - v0 , fx3 [ 2 ] , γ ) ;

--> r2 : resultant ( r1 , fx2 [ 2 ] , β ) ;

--> Gv : resultant ( r2 , fx1 [ 2 ] , α ) ;

--> Gv : factor ( Gv ) ;

--> pV : hipow ( Gv , v ) $
    if pV = NN then gv [ 0 ] : Gx else gv [ 0 ] : part ( Gv , 1 ) $
    gv [ 0 ] ;

--> /*ここ重要　拘束条件のリスト作成

    Const:[]$  巡回拡大の際に現れる2項方程式B[i]とその根として添加される数a[i]を
                巡回拡大が現れる度に拘束条件リストに付け加えられるリスト
              Const:cons([a[i],B[i]],const); という命令で積み重ねる

    CurConst:[]$　高速条件Consにその時点での最小多項式の条件を加えるた高速条件
                CurConst:cons([v,gv[i]],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*/ ;

--> Ω : ω ^ 2 + ω + 1 $
    Const : [ [ ω , Ω ] ] $
    CurConst : [ ] $
    CurConst : cons ( [ v , gv [ 0 ] ] , Const ) ;

--> fg0 : factor ( fx , gv [ 0 ] ) ;

--> AnsL : solve ( fg0 , x ) ;

--> an : [ ] $

--> for i : 1 thru 3 do ( an : endcons ( rhs ( AnsL [ i ] ) , an ) ) $
    an ;

--> sol : map ( lambda ( [ y , x ] , y = x ) , [ k [ 1 ] , k [ 2 ] , k [ 3 ] ] , an ) ;

--> kipL : listify ( permutations ( [ k [ 1 ] , k [ 2 ] , k [ 3 ] ] ) ) ;

--> V ( u , w , y ) : = u + 2 * w + 3 * y $

--> VpL : map ( lambda ( [ x ] , apply ( V , x ) ) , kipL ) ;
```

```
--> fix : 0 $
    for i : 1 thru NN do ( vkev : expand ( ev ( VpL [ i ] , sol ) ) , if vkev = v then fix : i ) $
    fix ;

--> abc : [ α , β , γ ] $
    aki : map ( lambda ( [ x , y ] , x = y ) , abc , kipL [ fix ] ) ;
    AnsL : ev ( aki , sol ) ;

--> VivL : expand ( ev ( VabcL , AnsL ) ) ;

--> check : [ ] $
    for i : 1 thru NN do (
      z : remainder ( subst ( rhs ( VivL [ i ] ) , v , gv [ 0 ] ) , gv [ 0 ] ) ,
      check : endcons ( z , check )
                    ) $
    check ;

--> /*  第2ステップ
       3次多項式のGvが因数分解がされ3次の多項式がgv[0]となった。
                従って ガロア群はGal0= [1,4,5]  のA3がガロア群G0と
                なることが分かった
                以下の計算は  Gal0=[1,4,5]から  Gal1=[1] への
                  計算である  Gal0/Gal1=3 である
     */

--> CurConst ;
    h0 : ev ( x - v [ 1 ] , VivL ) ;
    h1 : ev ( x - v [ 4 ] , VivL ) ;
    h2 : ev ( x - v [ 5 ] , VivL ) ;
    t0 : expand ( ( h0 + h1 + h2 ) / 3 ) $   t0 : Reduce ( t0 , CurConst ) ;

    t1 : expand ( ( h0 + ω * h1 + ω ^ 2 * h2 ) / 3 ) $ t1 : Reduce ( t1 , CurConst ) ;

    t2 : expand ( ( h0 + ω ^ 2 * h1 + ω * h2 ) / 3 ) $ t2 : Reduce ( t2 , CurConst ) ;


--> T1 : expand ( t1 ^ 3 ) $ T1 : Reduce ( T1 , CurConst ) ;
    t12 : t1 * t2 $ t12 : Reduce ( t12 , CurConst ) ;

--> A [ 1 ] : T1 $ B [ 1 ] : a [ 1 ] ^ 3 - A [ 1 ] ; t1 : a [ 1 ] $
    IA [ 1 ] : Inverse ( A [ 1 ] , CurConst ) ;

--> t2 : ( a [ 1 ] ^ 2 ) * t12 * IA [ 1 ] $ t2 : factor ( t2 ) ;

--> h0 : t0 + t1 + t2 $
    gx [ 1 ] : h0 ;
    gv [ 1 ] : subst ( v , x , h0 ) ;
```

```
--> gv[0];A[1];B[1];gv[1];

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

--> AnsL;

--> AL:[]$
    for i:1 thru 3 do(
       z:Reduce(rhs(AnsL[i]),CurConst),

     AL:endcons(z,AL)
    )$
    AL;

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

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

    rAL:ev(AL,Num)$  rAL:expand(rAL);

--> allroots(fx);

--> /*rAlとallroots(fx)を比較すると一致している。方程式の解き方が正しい*/
```
