```
> restart;
> DSCRMNT := proc(N :: integer, Res :: polynom) :: list;
  description "возвращет множество векторов для дискриминанта по правилу: для каждого
      момнома дискримината стпроится вектор из коэффифиента монома и показателей
      степеней каждой переменной в мономе"
  local i, ListOfTermsOfRes, ListOfListsOfDegrees, TermOfRes, ListOfDegrees;
  ListOfTermsOfRes := op(Res); #превращаем результант в список из его мономов
  #print(ListOfTermsOfRes);
  ListOfListsOfDegrees := [ ];
  for TermOfRes  in ListOfTermsOfRes do
      ListOfDegrees := [ ];
      for i from 0 to N  do
          ListOfDegrees := [degree(TermOfRes, a[i]), op(ListOfDegrees)];
      end do;
      ListOfDegrees := [coeffs(TermOfRes), op(ListOfDegrees)];
      ListOfDegrees := convert(ListOfDegrees, Vector);
      ListOfListsOfDegrees := [op(ListOfListsOfDegrees), ListOfDegrees]
  end do;
  #print(ListOfListsOfDegrees);
  return ListOfListsOfDegrees; #возвращаем список из векторов, где зашифрованы мономы
  end proc:

> MTRX := proc(n :: integer) :: Matrix;
  description "возвращает коэффициенты в левой части системы неравенств"
  local MTRX;
     MTRX := Matrix(n − 1, (j, k) → min(j, k)·(n − max(j, k)));
   return MTRX;
   end proc:

> CLMN :=proc(n :: integer) :: Matrix;
   description "возвращает столбец в правой части системы неравенств"
   local CLMN;
      CLMN := Matrix(n − 1, 1, k→n·k·(n − k));
   return CLMN;
   end proc:

> RSTVEC :=proc(v :: Vector) :: polynom;
   description
      "восстанавливает моном по вектору, где первая координата − это коэффициент монома, остальные −
   local x, i, RSTVEC;
   RSTVEC := v[1];
   i := 0;
   for x in v[2 ..] do
          RSTVEC := RSTVEC·a[i]^x;
          i := i + 1;
   end do;
   return RSTVEC;
   end proc:

> VCM := proc(a, b :: Matrix, N :: integer, signs :: Vector) :: integer;
```

```
description "возвращает единицу, если в векторах a и b равенство выполяется только в тех
    координатах, где в векторе signs стоят единицы, иначе возвращает ноль"
local i, VCM, aa, bb;
 aa := convert(a, Vector);
 bb := convert(b, Vector);
 VCM := 1;
   for i from 1 to N do
      if signs[i] = 1 and  aa[i] ≠ bb[i] then VCM := 0 end if;
   end do;
return VCM;
end proc:
```

```
> FACTORSR :=proc(N :: integer, facets :: list) :: polynom;
description "возвращает факторизацию срезки в терминах дискриминантов"
local truncation, i, numfacet, facets1, index, listofcoefficients, a, temp, j, polynom, b;
numfacet := nops(facets);
truncation := 1;
for i in facets do
     truncation := truncation · a[i]^2;
end do;
facets1 := [0, op(facets), N];
listofcoefficients := [ ];
for i from 0 to N do
     listofcoefficients := [op(listofcoefficients), a[i]]
 end do;
 for i from 1 to nops(facets1) − 1  do
     index := facets1[i + 1] − facets1[i];
     temp := op(facets1[i] + 1 ..facets1[i + 1] + 1, listofcoefficients);
     polynom := 0;
     j := 0 ;
     for b in temp do
          polynom := polynom + y^j · b;
          j := j + 1;
     end do;
     truncation := truncation·Δ[index](polynom);
 end do;
 return truncation;
 end proc:
```

```
> MAIN :=proc(N :: integer, facets :: list, nodegree :: list) :: polynom;
description "основная программа"
# N - это степень многочлена
# facets - это номера граней, на основе которых будет построен вектор signs
# nodegree - это номера пропущенных степеней, если все степени на месте, ставим пустое
    множество [ ]
uses LinearAlgebra;
local L, M, C, v, v1, signs, i, truncation, f, f1, Res;
signs := [ ];

     # signs - это вектор длины N-1 из нолей и единиц, НОЛЬ стоит там, где <=, ЕДИНИЦА
```

```
  - там, где =
for i from 1 to N − 1 do
    if i in facets
      then signs := [op(signs), 1]
      else signs := [op(signs), 0]
    end if;
end do;
signs := convert(signs, Vector);
f := 0;
#a[0] := 1; a[N] := 1; #нормируем многочлен, если необходимо
for i  from 0 to N do
   f := f + a[i]·y^i;
end do;
f1 := diff(f, y);
Res := simplify( resultant(f, f1, y) / a[N] · (−1)^(N·(N − 1)/2) );
print("МНОГОЧЛЕН :");
print(f); #печатаем многочлен
print("ПРОИЗВОДНАЯ : ");
print(f1);#печатаем его производную
print("ДИСКРИМИНАНТ : ");
print(Res);#печатаем его дискриминат - результант многочлена и его производной
    L := DSCRMNT(N, Res);
    #множество векторов из коэффифиента монома и показателей степенй всего
    дискриминанта
    M := MTRX(N); #коэфф левой части системы неравенств
   C := CLMN(N); #столбец в правой части системы неравенств
   # print(M, C); #на всякий случай напечатаем матрицу и столбец
truncation := 0;
#проверяем, удовлетворяют ли степени монома системе  неравенств
for v in L do
        v1 := v[3 ..−2];
        if  VCM(Multiply(M, v1), C, N − 1 , signs) = 1
          then truncation := truncation + RSTVEC(v);
        end if;
    end do;
print("СРЕЗКА НА НЕКООРДИНАТНЫЕ ГРАНИ: ");
print(truncation);
print("ФАКТОРИЗАЦИЯ СРЕЗКИ : ");
print( factor(truncation) );
print("ФАКТОРИЗАЦИЯ СРЕЗКИ В ТЕРМИНАХ ДИСКРИМИНАТОВ:");
print(FACTORSR(N, facets) );
if nodegree ≠ [ ]
then
  for i in nodegree do
      truncation := eval(truncation, a[i] = 0);
  end do;
print("ПРОИЗВОЛЬНАЯ СРЕЗКА (В ТОМ ЧИСЛЕ И НА КООРДИНАТНЫЕ ГРАНИ) :") ;
print( factor(truncation) );
end if;
#return factor(truncation) :
```

end proc:

> $MAIN(5, [2], [\ ]);$

"   : "

$$y^5 a_5 + y^4 a_4 + y^3 a_3 + y^2 a_2 + y a_1 + a_0$$

"   :   "

$$5 y^4 a_5 + 4 y^3 a_4 + 3 y^2 a_3 + 2 y a_2 + a_1$$

"   :   "

$$3125 a_0^4 a_5^4 - 2500 a_0^3 a_1 a_4 a_5^3 - 3750 a_0^3 a_2 a_3 a_5^3 + 2000 a_0^3 a_2 a_4^2 a_5^2 + 2250 a_0^3 a_3^2 a_4 a_5^2 - 1600 a_0^3 a_3 a_4^3 a_5 + 256 a_0^3 a_4^5 + 2000 a_0^2 a_1^2 a_3 a_5^3 - 50 a_0^2 a_1^2 a_4^2 a_5^2 + 2250 a_0^2 a_1 a_2^2 a_5^3 - 2050 a_0^2 a_1 a_2 a_3 a_4 a_5^2 + 160 a_0^2 a_1 a_2 a_4^3 a_5 - 900 a_0^2 a_1 a_3^3 a_5^2 + 1020 a_0^2 a_1 a_3^2 a_4^2 a_5 - 192 a_0^2 a_1 a_3 a_4^4 - 900 a_0^2 a_2^3 a_4 a_5^2 + 825 a_0^2 a_2^2 a_3^2 a_5^2 + 560 a_0^2 a_2^2 a_3 a_4^2 a_5 - 128 a_0^2 a_2^2 a_4^4 - 630 a_0^2 a_2 a_3^3 a_4 a_5 + 144 a_0^2 a_2 a_3^2 a_4^3 + 108 a_0^2 a_3^5 a_5 - 27 a_0^2 a_3^4 a_4^2 - 1600 a_0 a_1^3 a_2 a_5^3 + 160 a_0 a_1^3 a_3 a_4 a_5^2 - 36 a_0 a_1^3 a_4^3 a_5 + 1020 a_0 a_1^2 a_2^2 a_4 a_5^2 + 560 a_0 a_1^2 a_2 a_3^2 a_5^2 - 746 a_0 a_1^2 a_2 a_3 a_4^2 a_5 + 144 a_0 a_1^2 a_2 a_4^4 + 24 a_0 a_1^2 a_3^3 a_4 a_5 - 6 a_0 a_1^2 a_3^2 a_4^3 - 630 a_0 a_1 a_2^3 a_3 a_5^2 + 24 a_0 a_1 a_2^3 a_4^2 a_5 + 356 a_0 a_1 a_2^2 a_3^2 a_4 a_5 - 80 a_0 a_1 a_2^2 a_3 a_4^3 - 72 a_0 a_1 a_2 a_3^4 a_5 + 18 a_0 a_1 a_2 a_3^3 a_4^2 + 108 a_0 a_2^5 a_5^2 - 72 a_0 a_2^4 a_3 a_4 a_5 + 16 a_0 a_2^4 a_4^3 + 16 a_0 a_2^3 a_3^3 a_5 - 4 a_0 a_2^3 a_3^2 a_4^2 + 256 a_1^5 a_5^3 - 192 a_1^4 a_2 a_4 a_5^2 - 128 a_1^4 a_3^2 a_5^2 + 144 a_1^4 a_3 a_4^2 a_5 - 27 a_1^4 a_4^4 + 144 a_1^3 a_2^2 a_3 a_5^2 - 6 a_1^3 a_2^2 a_4^2 a_5 - 80 a_1^3 a_2 a_3^2 a_4 a_5 + 18 a_1^3 a_2 a_3 a_4^3 + 16 a_1^3 a_3^4 a_5 - 4 a_1^3 a_3^3 a_4^2 - 27 a_1^2 a_2^4 a_5^2 + 18 a_1^2 a_2^3 a_3 a_4 a_5 - 4 a_1^2 a_2^3 a_4^3 - 4 a_1^2 a_2^2 a_3^3 a_5 + a_1^2 a_2^2 a_3^2 a_4^2$$

"   :   "

$$108 a_0 a_2^5 a_5^2 - 72 a_0 a_2^4 a_3 a_4 a_5 + 16 a_0 a_2^4 a_4^3 + 16 a_0 a_2^3 a_3^3 a_5 - 4 a_0 a_2^3 a_3^2 a_4^2 - 27 a_1^2 a_2^4 a_5^2 + 18 a_1^2 a_2^3 a_3 a_4 a_5 - 4 a_1^2 a_2^3 a_4^3 - 4 a_1^2 a_2^2 a_3^3 a_5 + a_1^2 a_2^2 a_3^2 a_4^2$$

"   :   "

$$a_2^2 \left(27 a_2^2 a_5^2 - 18 a_2 a_3 a_4 a_5 + 4 a_2 a_4^3 + 4 a_3^3 a_5 - a_3^2 a_4^2\right) \left(4 a_0 a_2 - a_1^2\right)$$

"   : "

$$a_2^2 \Delta_2\left(y^2 a_2 + y a_1 + a_0\right) \Delta_3\left(y^3 a_5 + y^2 a_4 + y a_3 + a_2\right) \tag{1}$$