```
> with(NumberTheory):
```

```
>
```

Recovery of Exponents of Polynomials of High Degree

In what follows, procedures are defined wtih some examples. At the bottom of the worksheet I run the entire algorithm on random polynomials.

```
> # Procedures for generating random numbers and polynomials
```

```
> rand_int := proc(upper_bound) local R;
  description "randomly samples with replacement an integer from
  [0,upper_bound-1]";
  R := rand(upper_bound);
  return R();
  end proc:
```

```
>
```

This random polynomial procedure returns a list of the polynomials variables.

```
> rand_poly := proc(num_vars, coef_range, terms_upper_bound,
  total_degree_upper_bound)
  local i, var_list, total_degree,T;
  description "returns a random polynomial and a list of its variables";

  # create a list of variables
  var_list := [];
  for i from 1 to num_vars do
   var_list := [op(var_list), x||i];
  od;

  # use randpoly to generate polynomial
  return randpoly(var_list,coeffs=rand(-
  rand_int(coef_range)..rand_int(coef_range)), terms=terms_upper_bound,
  degree=total_degree_upper_bound),var_list;

  end proc:
```

```
>
```

```
>
```

```
> # An algorithm for generating smooth prime numbers (i.e. Algorithm 5 from the
  report)
```

Note, this algorithm should return FAIL if the primes are not found. I have not implemented this because the primes are generally always found.

```
> smooth_primes := proc(Delta,d)
  local primes, L, intger,k,i,j,n;
  description "";

  primes := []; i := 1; L:=1;

   while L<=d and i<169 do
    i := i+1;
    k := power_of_ith_prime(ithprime(i),Delta);
    n := floor(2^(63)/(Delta*ithprime(i)^k));

    while L<=d and n>1 do
     if isprime(Delta*ithprime(i)^k*n+1) then
      if not member(Delta*ithprime(i)^k*n+1,primes) then
       if ysmooth(Delta*ithprime(i)^k*n,1024) then
        primes := [op(primes),Delta*ithprime(i)^k*n+1];
        if nops(primes)>0 then L:= lcm(seq(primes[i]-1,i=1..nops(primes))); fi; fi;
  fi; fi;
     n := n-1;

    od;
   od;
  return primes;
  end:


>


> # subrutines for smooth_primes (i.e. for Algorithm 3)


> ysmooth := proc (n, y)
  local f, p;
  f := ifactors(n)[2];
  for p in f do
   if y < p[1] then return false fi;
  od;
  true;
  end proc:


> power_of_ith_prime := proc(p,Delta)
  local i;
  i := 1;
   while ((Delta)*(p^i) < 2^59) do i:= i+1; od;
  return i;
  end proc:
```

```
> generate_Delta := proc(terms) # this procedure generates a large enough delta
  given the number of terms
  local i;

  i := 3;
  while (2^i < terms^2) do i := i+1; od;

  return 2^i;
  end:
```

If Delta>t^2 then the probability that exponents are unique modulo Delta is greater than .5.

```
>
```

```
> # EXAMPLES of the smooth prime procedure
```

```
>
```

```
> LIST := smooth_primes(1000,10^1000)
```

$$LIST := [4323713773987629001, 7574089083986893001, 4078355660608327001, 8394795906194893001, 7749042374949132001, 8064600918940810001, 7258140827046729001, 5645220643258567001, 1324125243801622001, 7246594450667464001, 4010177473717516001, 5369821949545653001, 7053729832612479001, 8788821733194054001, 6982916358311716001, 8563014566487364001, 9207202486819605001, 5535649153405192001] \quad (1)$$

```
> for i from 1 to nops(LIST) do ifactor(LIST[i]-1); if not isprime(LIST[i]) then
  return("not prime!"); fi; od;
```

$$
\begin{gathered}
(2)^3\,(3)^{31}\,(5)^3\,(7) \\
(2)^3\,(5)^3\,(13)\,(17)^{12} \\
(2)^3\,(5)^3\,(7)\,(17)^{12} \\
(2)^3\,(5)^3\,(13)\,(71)^8 \\
(2)^5\,(3)\,(5)^3\,(71)^8 \\
(2)^4\,(5)^4\,(73)^8 \\
(2)^3\,(3)^2\,(5)^3\,(73)^8 \\
(2)^3\,(5)^3\,(7)\,(73)^8 \\
(2)^4\,(5)^3\,(131)^7 \\
(2)^6\,(5)^3\,(137)^7 \\
(2)^5\,(5)^3\,(139)^7 \\
(2)^3\,(3)\,(5)^3\,(151)^7 \\
(2)^3\,(3)\,(5)^3\,(157)^7 \\
(2)^4\,(3)\,(5)^3\,(337)^6 \\
(2)^5\,(5)^3\,(347)^6 \\
(2)^5\,(5)^3\,(359)^6 \\
(2)^3\,(3)\,(5)^4\,(907)^5 \\
(2)^6\,(5)^3\,(929)^5
\end{gathered} \tag{2}
$$

The primes are generated to be 1024-smooth.

```
> ################################

>

> # The Berlekamp Massey Algorithm

> BM := proc(s, N, P, x)
  local C,B,T,L,k,i,n,d,b,safemod,CC;
  ASSERT(nops(s) = 2*N);
  safemod := (exp, P) -> `if`(P=0, exp, exp mod P);
  B := 1;
  C := 1;
  L := 0;
  k := 1;
  b := 1;
  for n from 0 to 2*N-1 do
```

```
  d := s[n];
  for i from 1 to L do
   d := safemod(d + coeff(C,x^i)*s[n-i], P);
  od;
  if d=0 then k := k+1 fi;
  if (d <> 0 and 2*L > n) then
   C := safemod(expand(C - d*x^k*B/b), P);
   k := k+1;
  fi;
  if (d <> 0 and 2*L <= n) then
   T := C;
   C := safemod(expand(C - d*x^k*B/b), P);
   B := T;
   L := n+1-L;
   k := 1;
   b := d;

  fi;
 od:

 CC := coeff(C,x,N);
 for i from 1 to N do
  CC := CC + coeff(C,x,N-i)*x^i;
 od;

 return CC;

 end:
```

```
>
```

```
> # The procedure below interpolates exponents modulo p-1 for a list of primes
```

I print timings for evaluations, computation of the lambda polynomial (the Berlekamp Massy algorithm), factoring the lambda polynomial, and computation of the discrete logarithm.

```
> interp_exps_mod_p_minus_one := proc(f,p,T)
 local i,alpha, evaluations,V,S,L,Lambda,Lambda_Roots,E,v,Omega,st;

 # evaluations
 st := time();
 v := Array(0..2*T-1);
 alpha := PrimitiveRoot(p);
 Omega := Array(0..2*T-1);
 Omega[0] := 1;
 v[0] := Eval( f, {x=Omega[0]}) mod p;
 for i from 1 to 2*T-1 do
  Omega[i] := (Omega[i-1] mod p)*(alpha mod p) mod p;
  v[i] := Eval( f, {x=Omega[i]}) mod p; od;
 printf("Evaluations time=%10.4fs\n",time()-st);

 # compute lambda polynomial with the BM
 st := time();
 Lambda := BM(v,T,p,z);
```

```
printf("Solve time=%10.4fs\n",time()-st);

# computing roots
st := time();
Lambda_Roots := Roots(Lambda) mod p; # O(T^2 log p)
printf("Roots time=%10.4fs\n",time()-st);

st := time();
E := { seq( ModularLog( R[1],alpha,p ), R in Lambda_Roots ) };
printf("ModularLog time=%10.4fs\n",time()-st);

return E,v,alpha;
end proc:
```

```
>
```

Although I don't use it, I have a procedure for Rabins algorithm. Rabins algorithm splits a polynomial into linear factors.

```
> RABINS_ALGORITHM := proc(a,p)
  local g;
  g := Gcd(a, (Powmod(x,p,a,x) mod p) - x) mod p;
  return RABIN(g,p)
  end:
```

```
> RABIN := proc(f1,p)
  local f2,k,q;

  if f1=1 then return 1; fi;
  if degree(f1,x) = 1 then return f1; fi;
  f2 := 1; k := 1;
  while (f2=1 or f2=f1) and k<p do
   f2 := Gcd(f1, (Powmod(x+k,(p-1)/2,f1,x) mod p) - 1) mod p;
   k := k+1;
  od;

  q := Quo(f1,f2,x) mod p;
  return RABIN(f2,p)*RABIN(q,p);
  end:
```

```
> # EXAMPLE of rabins algorithm
```

```
> a := x^4 + 8*x^2 + 6*x + 8;
```

$$a := x^4 + 8x^2 + 6x + 8 \qquad (3)$$

```
> b := 11*x^4 + 8*x^2 + 6*x + 8;
```

$$b := 11x^4 + 8x^2 + 6x + 8 \qquad (4)$$

```
> RABINS_ALGORITHM(a,11)
```

$$(x+3)\,(x+1)\,(x+2)\,(x+5) \tag{5}$$

```
> Factor(a) mod 11
```

$$(x+3)\,(x+1)\,(x+2)\,(x+5) \tag{6}$$

```
> # since 11 divides the leading coefficient of b, 11 cannot be used with Rabins
  alorithm (we call 11 a bad prime).
```

```
> RABINS_ALGORITHM(b,11)
```

$$x+10 \tag{7}$$

```
> Factor(b) mod 11
```

$$8\,(x+10)^2 \tag{8}$$

Thus, 11 is a bad prime.

```
> # the generalized Chinese remainder theorem algorithm
```

```
> GCHREM := proc(U::list,m::list)
  local X,V,n,b,modulus,d,M;

  n := nops(U);

  if n=1 then
   #1.
   return U[1] mod m[1], m[1];

  else
   #2.
   V,M := GCHREM(U[1..nops(U)-1],m[1..nops(m)-1]);
   #3.
   d := igcdex(M,m[n],'t'); b := (U[n]-V)/d; modulus := m[n]/d;
   if (not type(b,integer)) then return FAIL,FAIL; fi;
   #4.
   X := t*b mod modulus;
   #5.
   return M*X + V, M*(m[n]/d);

  end if;
```

```
end proc:
```

This generalized Chinese remainder theorem procedure is similar to maples standard Chinese remainder theorem procedure. However, unlike maples chrem([],[]) procedure, GCHREM returns two values. The first value is the solution, the second value is the least common multiple of the moduli. The procedure returns FAIL if there is no solution.

```
> # EXAMPLES

>

> GCHREM([3,5],[6,10]);
```

$$15, 30 \tag{9}$$

```
> GCHREM([4,5],[6,10]);
```

$$FAIL, FAIL \tag{10}$$

```
> ##########################

> # procedures for applying and inverting the Kronecker substitution to a
  multivariate polynomial
```

The procedure returns both a univariate polynomial and the vector of substitution values. This vector is used to invert the Kronecker substitution.

```
> K_SUB := proc(P,vars)
  local r, M, P_function, univ_sub, i;

  r := []; # r is a list of variable degrees
  for i from 1 to nops(vars)-1 do
   r := [op(r), degree(P,vars[i])+1];
  od;

  M := [r[1]]; # M is a list of the degrees for the K-sub
  for i from 2 to nops(r) do
   M := [op(M),M[i-1]*r[i]];
  od;

  P_function := unapply(P,op(vars)); #

  univ_sub := [x]; # substitution variables
  for i from 1 to nops(vars)-1 do
   univ_sub := [op(univ_sub), x^M[i]];
  od;

  M := [1,op(M)];
```

```
return P_function(op(univ_sub)),M;
end proc:
```

```
>
```

```
> # EXAMPLE
```

```
> Describe(rand_poly)
```

```
          # returns a random polynomial and a list of its variables
rand_poly( num_vars, coef_range, terms_upper_bound, total_degree_upper_bound )
```

```
> f, vars := rand_poly(50,100,5,100)
```

$$f, vars := 62\, x1^{2}\, x2^{2}\, x3^{8}\, x7^{4}\, x8^{2}\, x9^{4}\, x10\, x11^{2}\, x12^{3}\, x13^{2}\, x14^{4}\, x15\, x16^{3}\, x17^{6}\, x18^{3}\, x21^{2}\, x23\, x24^{2}\, x25\, x28^{3}\, x29^{3}\, x30^{5}\, x35\, x38^{7}\, x39^{4}\, x40^{2}\, x41^{3}\, x48^{10}\, x49\, x50^{5} + 93\, x1^{6}\, x2\, x3\, x4^{2}\, x5^{3}\, x6^{2}\, x8^{4}\, x9^{6}\, x11^{6}\, x12\, x13^{3}\, x17\, x20^{7}\, x22^{4}\, x23^{4}\, x24^{4}\, x26\, x27^{9}\, x29\, x30\, x31^{3}\, x32^{4}\, x33^{3}\, x36\, x37\, x38\, x39\, x40^{2}\, x41^{4}\, x42\, x44\, x45^{2}\, x47^{3}\, x48\, x49^{3} + 33\, x2\, x3^{2}\, x4^{3}\, x5^{2}\, x6\, x7^{2}\, x8^{2}\, x10^{7}\, x11^{4}\, x12^{2}\, x14\, x15\, x16^{3}\, x17^{4}\, x19^{6}\, x21^{6}\, x22\, x23\, x24\, x26^{12}\, x27^{3}\, x29^{4}\, x32\, x34^{3}\, x37^{3}\, x39^{3}\, x41^{3}\, x44^{2}\, x46^{2}\, x48^{3}\, x49^{8}\, x50 + 85\, x2\, x3^{3}\, x4\, x5^{7}\, x7\, x9^{5}\, x10^{4}\, x11^{5}\, x12^{3}\, x13^{7}\, x15\, x17\, x21^{3}\, x24\, x25\, x26^{4}\, x27^{2}\, x29^{10}\, x30^{6}\, x31\, x32^{3}\, x33\, x34\, x35\, x36^{2}\, x37^{2}\, x38^{2}\, x39^{4}\, x40^{2}\, x41^{3}\, x42^{5}\, x47\, x48\, x50^{4} + 33\, x1^{8}\, x2^{2}\, x3^{3}\, x4^{2}\, x5\, x6\, x7^{6}\, x8\, x10\, x12^{2}\, x14^{2}\, x15^{2}\, x18\, x21\, x23^{4}\, x24^{2}\, x25^{2}\, x27^{2}\, x28^{3}\, x29^{6}\, x30^{2}\, x31^{3}\, x32^{5}\, x34^{3}\, x35^{3}\, x36^{2}\, x37^{2}\, x40^{6}\, x41^{2}\, x42\, x44^{2}\, x45^{13}\, x47\, x49\, x50^{2}, [x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17, x18, x19, x20, x21, x22, x23, x24, x25, x26, x27, x28, x29, x30, x31, x32, x33, x34, x35, x36, x37, x38, x39, x40, x41, x42, x43, x44, x45, x46, x47, x48, x49, x50] \tag{11}$$

```
> f_univariate, M := K_SUB(f,vars);
```

$$f\_univariate, M := 62\, x^{4077880155137233985243500565635635020} + 93\, x^{274739193297796366290751341689048820} + 33\, x^{1502894478189929358687945491701603440} + 85\, x^{3139427989480424176677040415326535265} + 33\, x^{1654321511294814557212356244284993725}, [1, 9, 27, 243, 972, 7776, 23328, 163296, 816480, 5715360, 45722880, 320060160, 1280240640, 10241925120, 51209625600, 153628876800, 614515507200, 4301608550400, 17206434201600, 120445039411200, 963560315289600, 6744922207027200, 33724611035136000, 168623055175680000, 843115275878400000, 2529345827635200000, 32881495759257600000, 328814957592576000000, 1315259830370304000000, 14467858134073344000000, 101275006938513408000000, 405100027754053632000000, 2430600166524321792000000, 9722400666097287168000000, 38889602664389148672000000, 155558410657556594688000000, 466675231972669784064000000, 1866700927890679136256000000, 14933607423125433090048000000, 74668037115627165450240000000, 522676259809390158151680000000, 2613381299046950790758400000000, 15680287794281704744550400000000, 15680287794281704744550400000000, 47040863382845114233651200000000, 658572087359831599271116800000000, 1975716262079494797813350400000000, 7902865048317979191253401600000000, 86931515531497771103787417600000000, 782383639783479939934086758400000000] \tag{12}$$

```
>
```

Notice the the univariate exponent growth! The next two procedures are for inverting the Kronecker substitution.

```
> # This is a subrutine
```

```
> KroneckerInv64s := proc( exponent::integer, M, n::integer )
  local j,d,y,X;
```

```
 y := exponent; X := Array(0..n-1);
 for j from n-1 by -1 to 1 do
  d := iquo(y , M[j]);
  X[j] := d;
  y := y - d*M[j]; # remainder
  od;
 X[0] := y;
  return X;
 end:
```

```
> # procedure for inverting the Kronecker substitution
```

```
> kronecker_inverse := proc(g,M,vars) # g = Kr(f)
 local i,j, monomials, exponents, f, array_monomial_powers, coeffs, term;

 monomials := [op(g)]; exponents := []; array_monomial_powers:= []; coeffs:=[];

 for i from 1 to nops(g) do
  exponents := [op(exponents),degree(monomials[i])];
  coeffs := [op(coeffs),coeff( monomials[i],x,exponents[i] ) ];
 od;

 f := 0;
 for i from 1 to nops(g) do
  array_monomial_powers := KroneckerInv64s(exponents[i],M,nops(M)+1);
  term := vars[1]^array_monomial_powers[1];
  for j from 2 to nops(vars) do term:=term*vars[j]^array_monomial_powers[j]; od;
  f := f +  coeffs[i]*term;
 od;

 return f;
 end:
```

```
> F := kronecker_inverse(f_univariate,M,vars);
```

$$F := 62\, x1^{2}\, x2^{2}\, x3^{8}\, x7^{4}\, x8^{2}\, x9^{4}\, x10\, x11^{2}\, x12^{3}\, x13^{2}\, x14^{4}\, x15\, x16^{3}\, x17^{6}\, x18^{3}\, x21^{2}\, x23\, x24^{2}\, x25\, x28^{3}\, x29^{3}\, x30^{5}\, x35\, x38^{7}\, x39^{4}\, x40^{2}\, x41^{3}\, x48^{10}\, x49\, x50^{5} + 93\, x1^{6}\, x2\, x3\, x4^{2}\, x5^{3}\, x6^{2}\, x8^{4}\, x9^{6}\, x11^{6}\, x12\, x13^{3}\, x17\, x20^{7}\, x22^{4}\, x23^{4}\, x24^{4}\, x26\, x27^{9}\, x29\, x30\, x31^{3}\, x32^{4}\, x33^{3}\, x36\, x37\, x38\, x39\, x40^{2}\, x41^{4}\, x42\, x44\, x45^{2}\, x47^{3}\, x48\, x49^{3} + 33\, x2\, x3^{2}\, x4^{3}\, x5^{2}\, x6\, x7^{2}\, x8^{2}\, x10^{7}\, x11^{4}\, x12^{2}\, x14\, x15\, x16^{3}\, x17^{4}\, x19^{6}\, x21^{6}\, x22\, x23\, x24\, x26^{12}\, x27^{3}\, x29^{4}\, x32\, x34^{3}\, x37^{3}\, x39^{3}\, x41^{3}\, x44^{2}\, x46^{2}\, x48^{3}\, x49^{8}\, x50 + 85\, x2\, x3^{3}\, x4\, x5^{7}\, x7\, x9^{5}\, x10^{4}\, x11^{5}\, x12^{3}\, x13^{7}\, x15\, x17\, x21^{3}\, x24\, x25\, x26^{4}\, x27^{2}\, x29^{10}\, x30^{6}\, x31\, x32^{3}\, x33\, x34\, x35\, x36^{2}\, x37^{2}\, x38^{2}\, x39^{4}\, x40^{2}\, x41^{3}\, x42^{5}\, x47\, x48\, x50^{4} + 33\, x1^{8}\, x2^{2}\, x3^{3}\, x4^{2}\, x5\, x6\, x7^{6}\, x8\, x10\, x12^{2}\, x14^{2}\, x15^{2}\, x18\, x21\, x23^{4}\, x24^{2}\, x25^{2}\, x27^{2}\, x28^{3}\, x29^{6}\, x30^{2}\, x31^{3}\, x32^{5}\, x34^{3}\, x35^{3}\, x36^{2}\, x37^{2}\, x40^{6}\, x41^{2}\, x42\, x44^{2}\, x45^{13}\, x47\, x49\, x50^{2} \tag{13}$$

```
> evalb(F=f);
```

$$true \tag{14}$$

```
> # additional procedures
```

```
> # sorting exponents modulo Delta

> BS_Array := proc(S::set,g::integer)
  local i,j,store,S_A;
  #description "sorts a set modulo Delta into a list";
  S_A := Array(1..nops(S));
  for i from 1 to nops(S) do S_A[i] := S[i]; od;
  for i from 1 to nops(S)-1 do
   for j from 1 to nops(S)-1 do
    if S_A[j] mod g <S_A[j+1] mod g then store:=S_A[j]; S_A[j]:=S_A[j+1];
  S_A[j+1]:=store; fi;
   od;
  od;

  return S_A;
  end proc:

> # extracting univariate exponents from a polynomial

> univar_poly_exponents_Array := proc(f)
  local i,exp_array,monomial_set,t;

  description "puts exponents of a polynomial into an array in numerically sorted
  order";
  t := nops(f);
  monomial_set := {op(f)}; exp_array := Array(1..t);

  for i from 1 to t do exp_array[i] := degree(monomial_set[i]); od;

  return exp_array;
  end proc:

> ################################################################

> 

> ################################################################

> 

> ################################################################

> 

> 
```

```
>

>

>

>

>

>

>

>

>

>

>

>

>

> # The main algorithm (i.e. algorithm 4)
```

Note: if Delta > t^2 then the probability that exponents are unique is greater than .5. I have taken Delta 15 times larger than this so that exponents are unique most of the time. The algorithm should pick a new Delta and try again if the exponents are not unique.

```
>

> HIGH_DEGREE_POLYNOMIAL_INTERPOLATION := proc(P,vars)
  local i,Delta, t,P_univ,M,exponents,d,prime_list,s,E,v,alpha,OUTPUT,st;

  st := time():
```

```
t := nops(P); # assumeing the number of terms is known
Delta := generate_Delta(t); Delta := 15*Delta; # increasing the probability that
exponents are unique
P_univ,M := K_SUB(P,vars);
exponents := univar_poly_exponents_Array(P_univ): # for testing - this set is
sorted numberically
d := max(exponents); # assumeing that d is known

prime_list := smooth_primes(Delta,d);
s := nops(prime_list);
for i from 1 to s do
 E||i,v,alpha := interp_exps_mod_p_minus_one(P_univ,prime_list[i],t)
od:

# remeber, since we can evaluate the multivariate polynomial, we can also
evaluate the
# univariate polynomial by applying a substitution.

if (nops(E1 mod Delta) <> nops(E1)) then error print("Exponents are not unique
modulo Delta"); fi;

for i from 1 to s do E_sorted||i := BS_Array(E||i,Delta); od: # sorting modulo
Delta
E := Array(1..t); # constructing the solution
for i from 1 to t do
 E[i] := GCHREM([seq(E_sorted||j[i],j=1..s)],[seq(prime_list[j]-1,j=1..s)])[1];
od:

OUTPUT := mod_coeff_interp(E, v, prime_list, alpha, t, P_univ, M); # coefficient
recovery
return OUTPUT;
end:
```

```
>
```

mod_coeff_interp is short for modular coefficient interpolation. This procedure is used in the one above.

```
> mod_coeff_interp := proc(E,v,prime_list,alpha,T,P_univ,M)
  local p, monomials, i, TVM, V, coefficients, F, P_inv, st;

   p := prime_list[1];
   monomials := Array(1..T);

   for i from 1 to T do monomials[i] := alpha &^ (E[i] mod (p-1)) mod p; od;
   TVM :=
  Matrix([seq([seq(monomials[i]^j,i=1..ArrayNumElems(monomials))],j=0..T-1)]) mod
  p;
   V := Vector([seq(v[i],i=0..T-1)]);
   `mod` := mods;
   coefficients := Linsolve(TVM,V) mod p;

   F := 0;
   for i from 1 to T do
    F := F + coefficients[i]*x^E[i];
   od:
   P_inv := kronecker_inverse(P_univ,M,vars);
```

```
 return "constructed Polynomial = original polynomial: " * evalb(P_inv=P);
 end:
```

```
> # generating random polynomials and running the algorithm:
```

```
> Describe(rand_poly)
```

```
          # returns a random polynomial and a list of its variables
rand_poly( num_vars, coef_range, terms_upper_bound, total_degree_upper_bound )
```

```
> P,vars := rand_poly(10,10,50,10):
```

```
>
```

```
> HIGH_DEGREE_POLYNOMIAL_INTERPOLATION(P,vars);
```

```
Evaluations time=    0.0110s

Solve time=    0.0780s
Roots time=    0.0630s

ModularLog time=    0.2250s
```

"constructed Polynomial = original polynomial: " *true* (15)

```
> Describe(rand_poly)
```

```
          # returns a random polynomial and a list of its variables
rand_poly( num_vars, coef_range, terms_upper_bound, total_degree_upper_bound )
```

```
> P,vars := rand_poly(10,10,10,10);
```

$$P, vars := -x1^4\,x10^3\,x4^2\,x5 + 3\,x1^2\,x10^3\,x2\,x3\,x4\,x5\,x6 + 2\,x1^2\,x10\,x2^2\,x4\,x6\,x8^3 + 3\,x1^2\,x2^4\,x3\,x4\,x6\,x9 + x10^3\,x2^6\,x6 - x10\,x3\,x5\,x6^4\,x8^3 - x3\,x5^4\,x6^3\,x8^2 + 3\,x1^2\,x10\,x2\,x3\,x6\,x7\,x8\,x9 + 2\,x6\,x7^4\,x8\,x9^3 + 3\,x1\,x4\,x5^4\,x6, [x1, x2, x3, x4, x5, x6, x7, x8, x9, x10] \quad (16)$$

```
> HIGH_DEGREE_POLYNOMIAL_INTERPOLATION(P,vars);
```

```
Evaluations time=    0.0010s

Solve time=    0.0030s
Roots time=    0.0090s
ModularLog time=    0.1250s
```

"constructed Polynomial = original polynomial: " *true* (17)

```
> Describe(rand_poly)
```

```
          # returns a random polynomial and a list of its variables
rand_poly( num_vars, coef_range, terms_upper_bound, total_degree_upper_bound )
```

```
> P,vars := rand_poly(100,100,100,1000):

>

> HIGH_DEGREE_POLYNOMIAL_INTERPOLATION(P,vars);
```

```
Evaluations time=    0.2050s

Solve time=    0.0810s

Roots time=    0.1680s

ModularLog time=    0.5120s

Evaluations time=    0.2190s

Solve time=    0.0930s

Roots time=    0.1840s
```

```
ModularLog time=    0.5590s

Evaluations time=    0.2140s

Solve time=    0.1000s

Roots time=    0.1910s

ModularLog time=    0.5790s

Evaluations time=    0.2170s
Solve time=    0.0830s

Roots time=    0.1810s

ModularLog time=    0.5270s

Evaluations time=    0.2310s

Solve time=    0.0870s

Roots time=    0.2570s

ModularLog time=    0.4510s

Evaluations time=    0.2240s

Solve time=    0.0890s

Roots time=    0.2700s

ModularLog time=    0.4110s

Evaluations time=    0.2250s

Solve time=    0.1680s
```

```
      Roots time=    0.1960s

 ModularLog time=    0.4030s

Evaluations time=    0.2370s

      Solve time=    0.1570s

      Roots time=    0.1810s

 ModularLog time=    0.4170s

Evaluations time=    0.2850s

      Solve time=    0.0850s
      Roots time=    0.1800s

 ModularLog time=    0.4270s

Evaluations time=    0.2980s
      Solve time=    0.0850s

      Roots time=    0.1880s

 ModularLog time=    0.3960s

Evaluations time=    0.2990s
      Solve time=    0.0770s

      Roots time=    0.1910s

 ModularLog time=    0.4600s

Evaluations time=    0.3400s
      Solve time=    0.0800s

      Roots time=    0.1790s
```

```
ModularLog time=    0.4590s

Evaluations time=    0.3120s
Solve time=    0.1020s

Roots time=    0.1780s

ModularLog time=    0.5220s

Evaluations time=    0.2490s

Solve time=    0.0870s

Roots time=    0.2770s

ModularLog time=    0.6280s

Evaluations time=    0.3260s

Solve time=    0.1280s

Roots time=    0.2830s

ModularLog time=    0.6490s

Evaluations time=    0.2520s
Solve time=    0.1140s

Roots time=    0.2200s

ModularLog time=    0.6200s

Evaluations time=    0.2380s

Solve time=    0.1090s

Roots time=    0.2090s
```

```
ModularLog time=    0.6090s
```

"constructed Polynomial = original polynomial: " *true* (18)

```
> Describe(rand_poly)
```

```
          # returns a random polynomial and a list of its variables
rand_poly( num_vars, coef_range, terms_upper_bound, total_degree_upper_bound )
```

```
> P,vars := rand_poly(10,10,10,1000);
```

$$P, vars := x1^{53}\, x2^{46}\, x3^{7}\, x4^{95}\, x5^{28}\, x6^{30}\, x7^{204}\, x8^{11}\, x9^{70}\, x10^{197} + x1^{129}\, x2^{18}\, x3\, x4^{268}\, x5^{3}\, x6^{26}\, x7^{46}\, x8^{85}\, x9^{204}\, x10^{125} - x1^{67}\, x2^{167}\, x3^{35}\, x4^{69}\, x5^{15}\, x6^{40}\, x7^{500}\, x8^{12}\, x9^{12}\, x10^{6} - x1^{146}\, x2^{99}\, x3^{232}\, x4\, x5^{43}\, x6^{118}\, x7^{20}\, x8^{105}\, x9^{56}\, x10^{152} - x1^{102}\, x2^{17}\, x3^{114}\, x4^{62}\, x5^{37}\, x6^{311}\, x7^{5}\, x8^{72}\, x9^{246}\, x10^{15}, [x1, x2, x3, x4, x5, x6, x7, x8, x9, x10] \quad (19)$$

```
>
```

```
> HIGH_DEGREE_POLYNOMIAL_INTERPOLATION(P,vars);
```

```
Evaluations time=    0.0020s

     Solve time=    0.0010s
     Roots time=    0.0020s
ModularLog time=    0.0340s
Evaluations time=    0.0010s
     Solve time=    0.0000s
     Roots time=    0.0010s
ModularLog time=    0.0230s
Evaluations time=    0.0010s
     Solve time=    0.0010s
     Roots time=    0.0020s
ModularLog time=    0.0190s
```

"constructed Polynomial = original polynomial: " *true* (20)

```
> Describe(rand_poly)
```

```
          # returns a random polynomial and a list of its variables
rand_poly( num_vars, coef_range, terms_upper_bound, total_degree_upper_bound )
```

```
> P,vars := rand_poly(200,10,10,500);
```

$$
\begin{gathered}
P, \\
vars := 6\, x108^{5}\, x109\, x111\, x112\, x114^{2}\, x116^{4}\, x118^{4}\, x119\, x120^{5}\, x123^{2}\, x125^{2}\, x126^{3}\, x127\, x128^{3}\, x129^{3}\, x130^{3}\, x132\, x134^{7}\, x135^{6}\, x136^{4}\, x138\, x139^{6}\, x142^{5}\, x143^{2} \\
x144^{8}\, x145^{8}\, x147\, x193^{3}\, x196\, x197\, x198^{2}\, x199\, x150^{2}\, x151\, x152^{8}\, x154^{2}\, x155^{2}\, x157^{6}\, x158^{4}\, x159^{11}\, x160\, x161^{7}\, x162^{3}\, x163^{4}\, x164^{3}\, x165^{2}\, x166\, x167\, x170\, x171 \\
x172\, x174^{4}\, x178^{6}\, x180\, x181^{8}\, x183^{18}\, x184^{3}\, x185\, x188\, x189^{3}\, x190\, x191^{5}\, x192\, x16^{5}\, x17^{12}\, x18^{4}\, x20^{8}\, x22^{3}\, x23\, x25^{4}\, x27^{6}\, x28^{4}\, x30^{7}\, x33\, x35^{4}\, x40^{3}\, x42\, x44\, x45^{2} \\
x48^{4}\, x49^{2}\, x1^{7}\, x2^{2}\, x6\, x9^{2}\, x14^{4}\, x15^{2}\, x102^{3}\, x104\, x106\, x107^{5}\, x82^{2}\, x83^{6}\, x87^{9}\, x89^{7}\, x90^{2}\, x91\, x93^{3}\, x94^{5}\, x95^{4}\, x96^{4}\, x98^{7}\, x99^{3}\, x100^{2}\, x43^{21}\, x51\, x52^{3}\, x53\, x55\, x56^{2}\, x58^{3} \\
x59^{2}\, x60\, x61^{3}\, x62^{2}\, x63^{13}\, x64^{4}\, x65^{5}\, x67\, x68^{8}\, x69^{4}\, x70^{13}\, x72^{4}\, x73^{17}\, x74^{5}\, x75\, x76^{2}\, x77^{4}\, x78^{2}\, x79^{2}\, x80^{3}\, x81 \\
-3\, x110^{5}\, x111\, x113^{4}\, x115^{7}\, x117^{7}\, x118\, x119^{4}\, x121^{6}\, x124^{4}\, x125^{9}\, x127\, x128^{4}\, x129^{2}\, x130\, x131^{2}\, x132\, x134\, x135^{8}\, x136^{2}\, x138^{6}\, x139\, x140\, x142^{5}\, x144^{5}\, x146 \\
x147\, x149\, x193\, x194^{3}\, x195^{3}\, x196^{3}\, x152^{4}\, x153\, x154^{5}\, x156^{2}\, x157^{7}\, x158^{2}\, x159^{2}\, x160^{6}\, x161^{2}\, x163\, x164^{5}\, x165^{6}\, x167^{3}\, x168^{3}\, x169^{2}\, x170^{3}\, x171^{2}\, x172\, x173^{5} \\
x174^{2}\, x175^{3}\, x176^{7}\, x177^{3}\, x178^{8}\, x179^{3}\, x180^{3}\, x182\, x183^{2}\, x184^{2}\, x185\, x186\, x187\, x188^{9}\, x192^{13}\, x16^{2}\, x17\, x19^{6}\, x20^{9}\, x22^{4}\, x23\, x24^{3}\, x26^{3}\, x27\, x28^{4}\, x30^{8}\, x31^{10}\, x32^{7} \\
x33^{4}\, x34^{7}\, x35^{4}\, x39^{7}\, x40^{6}\, x42^{3}\, x45^{6}\, x46^{2}\, x47\, x48^{2}\, x1^{2}\, x2^{3}\, x4^{3}\, x5^{5}\, x6^{2}\, x7^{5}\, x8^{6}\, x9^{2}\, x10^{3}\, x12^{4}\, x13^{5}\, x14^{4}\, x15^{4}\, x101^{2}\, x104^{2}\, x105^{9}\, x106^{6}\, x107^{6}\, x83\, x84^{2}\, x85^{6}\, x87 \\
x88^{4}\, x89^{6}\, x90^{7}\, x93\, x94^{2}\, x98^{3}\, x99\, x100^{2}\, x43\, x51^{3}\, x52\, x53^{2}\, x54^{4}\, x55^{2}\, x56^{3}\, x57\, x61^{2}\, x65^{2}\, x66\, x67^{4}\, x69^{4}\, x71^{6}\, x72^{8}\, x73^{8}\, x74^{2}\, x75^{2}\, x76^{6}\, x79^{2}\, x80^{2} \\
+2\, x108\, x109\, x110\, x112^{2}\, x113^{3}\, x114\, x115^{15}\, x116^{2}\, x117^{2}\, x119^{2}\, x120^{3}\, x121\, x122^{3}\, x123^{8}\, x124\, x126\, x127^{9}\, x128^{5}\, x130\, x132^{3}\, x133^{3}\, x134\, x135^{5}\, x136\, x138^{3} \\
x139^{2}\, x140^{4}\, x142^{6}\, x143^{4}\, x144^{3}\, x145\, x146\, x148\, x194^{2}\, x195\, x196\, x197^{5}\, x198^{5}\, x199^{2}\, x200^{2}\, x150^{10}\, x151\, x152^{4}\, x153^{3}\, x154\, x156\, x157^{3}\, x158^{4}\, x159^{9}\, x160^{2}\, x161 \\
x163^{3}\, x166^{4}\, x167^{5}\, x169\, x171^{4}\, x173^{5}\, x174^{4}\, x175^{4}\, x176^{3}\, x177^{5}\, x178\, x180^{3}\, x181\, x183^{3}\, x184\, x185\, x186^{2}\, x188^{3}\, x191^{4}\, x192^{14}\, x16^{3}\, x17^{10}\, x19^{4}\, x20^{3}\, x21\, x23^{5} \\
x24^{3}\, x25^{9}\, x27^{3}\, x28\, x29^{2}\, x30^{8}\, x32^{3}\, x33^{6}\, x34\, x35^{2}\, x36^{11}\, x37\, x38^{4}\, x39^{3}\, x40^{2}\, x42^{5}\, x44\, x45\, x46^{2}\, x49\, x50^{2}\, x1^{7}\, x2^{9}\, x4^{3}\, x5^{4}\, x6\, x7^{2}\, x8^{4}\, x9^{3}\, x10\, x11^{3}\, x12^{3}\, x13\, x14^{2} \\
x101^{3}\, x103\, x104^{3}\, x105^{4}\, x106^{2}\, x82\, x83^{3}\, x84^{3}\, x86^{11}\, x89\, x90^{6}\, x91\, x93\, x95^{2}\, x96^{3}\, x97\, x98^{6}\, x99^{7}\, x100\, x43\, x51^{11}\, x55^{4}\, x57\, x58\, x59^{7}\, x60^{2}\, x63^{3}\, x64^{3}\, x66\, x68\, x70^{2} \\
x71^{11}\, x72^{7}\, x73^{2}\, x74\, x76\, x77^{7}\, x78^{2}\, x79\, x80\, x81 \\
-2\, x108^{5}\, x109^{2}\, x112\, x113^{5}\, x115\, x116^{6}\, x117\, x118\, x119^{3}\, x120^{15}\, x121^{2}\, x122^{3}\, x123^{6}\, x124^{3}\, x125^{4}\, x127^{2}\, x128^{6}\, x130\, x132^{2}\, x133^{10}\, x134^{2}\, x135^{2}\, x138^{6}\, x139 \\
x140^{2}\, x142\, x146^{4}\, x147\, x148\, x194^{3}\, x195\, x197^{5}\, x198^{2}\, x199^{2}\, x200^{3}\, x150\, x151^{2}\, x152^{4}\, x153^{2}\, x154^{8}\, x155\, x156\, x158^{6}\, x160^{9}\, x161^{6}\, x163^{10}\, x164\, x165^{3}\, x166^{2} \\
x167^{3}\, x168^{2}\, x169^{2}\, x171^{3}\, x173^{3}\, x174^{4}\, x175^{4}\, x176^{6}\, x177^{18}\, x178^{2}\, x179^{2}\, x180\, x182\, x183^{2}\, x185^{3}\, x187^{4}\, x188^{2}\, x190^{3}\, x191\, x192^{4}\, x16\, x17^{8}\, x19\, x20^{2}\, x21^{8}\, x22^{3} \\
x25^{2}\, x26^{5}\, x27^{5}\, x30^{4}\, x31^{7}\, x32\, x33^{5}\, x35^{6}\, x36^{11}\, x38\, x39^{3}\, x40^{7}\, x41^{5}\, x44\, x46^{13}\, x48^{5}\, x49^{3}\, x50\, x1^{7}\, x3^{4}\, x5^{8}\, x8^{6}\, x9\, x11^{3}\, x12\, x13^{2}\, x15\, x101^{3}\, x102^{9}\, x104^{4}\, x105\, x106 \\
x107^{15}\, x82^{4}\, x83\, x84^{6}\, x85^{2}\, x86^{2}\, x89^{2}\, x91^{2}\, x92\, x93^{7}\, x94^{3}\, x95^{2}\, x96^{2}\, x97^{2}\, x98^{3}\, x99^{5}\, x100\, x43^{2}\, x53\, x54^{2}\, x55^{2}\, x57^{5}\, x58^{4}\, x61^{2}\, x62^{2}\, x63\, x64\, x65\, x66\, x70\, x71^{2}\, x74 \\
x75^{2}\, x76\, x77^{4}\, x78^{2}\, x80^{2} \\
+7\, x108^{2}\, x110\, x111^{2}\, x112\, x113\, x117^{14}\, x118^{12}\, x119^{2}\, x120\, x123^{4}\, x128^{2}\, x129^{2}\, x130^{2}\, x131^{3}\, x132\, x134\, x136^{5}\, x139\, x140^{2}\, x141^{3}\, x143\, x144^{2}\, x145^{4}\, x146^{11} \\
x147^{4}\, x149\, x193^{10}\, x194^{3}\, x198\, x200^{3}\, x150^{8}\, x152^{2}\, x153^{3}\, x154\, x156^{3}\, x160\, x161\, x163^{5}\, x164^{2}\, x167^{5}\, x169^{6}\, x170^{2}\, x171\, x172\, x173^{2}\, x175^{3}\, x176^{2}\, x178^{2}\, x181^{2} \\
x182^{2}\, x183^{2}\, x184\, x185^{4}\, x186^{2}\, x187^{11}\, x188^{2}\, x190^{7}\, x192^{2}\, x16^{2}\, x18\, x19^{4}\, x20\, x21^{4}\, x22^{3}\, x23\, x24^{12}\, x25\, x26\, x27^{5}\, x28^{4}\, x29\, x30^{4}\, x31^{6}\, x33^{3}\, x34^{10}\, x35\, x36^{3}\, x37 \\
x38^{4}\, x39^{5}\, x40^{7}\, x41^{2}\, x46^{7}\, x48^{7}\, x49\, x1^{2}\, x3^{5}\, x4^{11}\, x5^{5}\, x7^{3}\, x10^{8}\, x12^{23}\, x14\, x101^{6}\, x102^{2}\, x103^{4}\, x104\, x106^{3}\, x82^{4}\, x83^{10}\, x84^{2}\, x85^{4}\, x88^{3}\, x89\, x90\, x92\, x93^{10}\, x95\, x96 \\
x98^{5}\, x99^{2}\, x43^{10}\, x51^{2}\, x52^{2}\, x53^{2}\, x54^{6}\, x56^{3}\, x57^{4}\, x61^{9}\, x62^{4}\, x63^{9}\, x65^{3}\, x66^{4}\, x67\, x70^{2}\, x71^{5}\, x72\, x74^{2}\, x76^{7}\, x78^{8}\, x80^{6}\, x81^{2} \\
-3\, x108^{3}\, x109^{6}\, x110^{5}\, x111\, x112^{2}\, x113^{7}\, x116\, x117^{5}\, x118\, x120\, x121^{6}\, x123^{6}\, x124^{10}\, x125^{8}\, x126\, x127\, x128\, x129\, x130\, x131\, x132^{8}\, x134^{5}\, x135^{4}\, x136^{2}\, x138 \\
x139^{2}\, x142\, x143^{4}\, x145\, x147^{2}\, x193\, x197\, x200^{3}\, x152^{4}\, x154^{5}\, x155^{3}\, x156\, x157^{3}\, x158\, x159^{3}\, x161^{3}\, x162^{2}\, x163^{3}\, x164\, x165^{2}\, x166^{4}\, x167^{2}\, x168^{12}\, x169^{3}\, x170^{9} \\
x171^{2}\, x173\, x174\, x176^{10}\, x178^{13}\, x180\, x181^{5}\, x182^{2}\, x183^{2}\, x184^{5}\, x185\, x186\, x187^{2}\, x188\, x189\, x192\, x16^{2}\, x19^{4}\, x20^{3}\, x21^{4}\, x22^{2}\, x23^{5}\, x25\, x26^{4}\, x27^{2}\, x28^{5}\, x29\, x30 \\
x31^{7}\, x32^{2}\, x33^{3}\, x35^{2}\, x37^{2}\, x38^{6}\, x39^{2}\, x40\, x41^{2}\, x42^{2}\, x44^{2}\, x45^{5}\, x46^{12}\, x47^{9}\, x49^{2}\, x50^{4}\, x1\, x2^{3}\, x3^{2}\, x4^{7}\, x5\, x6^{4}\, x7^{2}\, x8^{2}\, x9^{2}\, x10\, x11\, x12^{5}\, x13^{4}\, x15^{2}\, x101^{6}\, x102^{2}\, x103^{2} \\
x104\, x105^{3}\, x106^{8}\, x107^{2}\, x84^{2}\, x86^{8}\, x87^{2}\, x88^{10}\, x89\, x92^{3}\, x93^{7}\, x94^{2}\, x95^{3}\, x97\, x43^{2}\, x53^{3}\, x54^{10}\, x55^{2}\, x56^{7}\, x57\, x58^{2}\, x59^{4}\, x60\, x61^{7}\, x62\, x65^{3}\, x66^{2}\, x67^{3}\, x69^{12}\, x70^{3} \\
x71^{2}\, x73^{3}\, x75^{3}\, x76^{4}\, x77\, x78^{3}\, x79^{2}\, x80^{3}\, x81^{4} \\
+6\, x108^{5}\, x110\, x113\, x115^{4}\, x116^{2}\, x118\, x120\, x121^{3}\, x122\, x123\, x125\, x126^{4}\, x127^{8}\, x128^{3}\, x130\, x131^{4}\, x133\, x134^{9}\, x135^{2}\, x136^{3}\, x138^{3}\, x143^{4}\, x144^{8}\, x145^{2}\, x146^{2} \\
x148^{24}\, x193^{4}\, x194^{12}\, x197^{2}\, x199^{3}\, x200\, x150^{2}\, x152^{2}\, x153^{2}\, x154^{13}\, x156\, x158\, x159\, x160^{8}\, x161^{4}\, x162^{2}\, x163^{5}\, x164^{15}\, x165\, x166^{3}\, x168^{2}\, x170\, x172^{7}\, x173^{4}\, x175^{3} \\
x176^{2}\, x178^{4}\, x179^{6}\, x180^{11}\, x182\, x183\, x184\, x185\, x187^{9}\, x190^{3}\, x191^{3}\, x192^{7}\, x16^{4}\, x17\, x18^{6}\, x19^{2}\, x20\, x22^{2}\, x23^{3}\, x24^{2}\, x26\, x28^{2}\, x30^{2}\, x31^{5}\, x32^{3}\, x34^{3}\, x35\, x36\, x37^{3} \\
x38\, x40^{3}\, x42^{2}\, x44^{4}\, x45\, x47^{2}\, x48^{6}\, x49^{7}\, x50^{8}\, x1^{4}\, x2^{3}\, x3\, x4^{3}\, x5\, x6^{2}\, x8\, x9^{3}\, x11^{4}\, x14^{6}\, x15\, x101^{4}\, x102^{4}\, x103^{2}\, x104\, x105^{6}\, x106^{2}\, x107\, x82^{2}\, x83^{2}\, x85^{2}\, x86\, x87\, x88^{6} \\
x89^{4}\, x90^{6}\, x91\, x92^{5}\, x94\, x95\, x96^{3}\, x97^{9}\, x98^{3}\, x99^{2}\, x100^{7}\, x43^{2}\, x51^{2}\, x52^{8}\, x53^{5}\, x56^{7}\, x58^{2}\, x59\, x60\, x61^{3}\, x62^{3}\, x64^{3}\, x65^{5}\, x66^{2}\, x70^{2}\, x71^{5}\, x72\, x73^{2}\, x74\, x76^{2}\, x77^{2}\, x78^{3} \\
x79^{3}\, x80^{4}\, x81^{3} \\
+x109^{2}\, x111^{2}\, x113^{2}\, x114^{11}\, x116^{6}\, x117^{4}\, x118^{5}\, x119^{7}\, x120\, x124^{2}\, x125^{3}\, x126^{3}\, x127^{6}\, x128\, x129\, x131^{5}\, x132\, x133^{3}\, x134^{2}\, x135^{5}\, x136^{3}\, x137^{2}\, x138^{4}\, x139^{10} \\
x140^{2}\, x141\, x142^{5}\, x143\, x144^{3}\, x145\, x146^{2}\, x147\, x148^{4}\, x193^{2}\, x195^{2}\, x196^{3}\, x197\, x198^{3}\, x199^{5}\, x200\, x150\, x151^{3}\, x152^{2}\, x154^{7}\, x155^{2}\, x157\, x158^{3}\, x159^{2}\, x161^{2}\, x162 \\
x163^{3}\, x164\, x165^{10}\, x166\, x167^{3}\, x168^{2}\, x170^{3}\, x172\, x173^{3}\, x174^{2}\, x175^{2}\, x177^{2}\, x178^{2}\, x179^{2}\, x180^{7}\, x182\, x183^{7}\, x184^{3}\, x188^{9}\, x190^{7}\, x192^{13}\, x16\, x18^{2}\, x19^{8}\, x20^{2}\, x21 \\
x22\, x23^{2}\, x28^{3}\, x29^{2}\, x31^{5}\, x32^{5}\, x33^{4}\, x34^{4}\, x36^{4}\, x37^{6}\, x38^{8}\, x40^{5}\, x42\, x44\, x45^{10}\, x46^{2}\, x48^{8}\, x49^{8}\, x1^{2}\, x2\, x3^{10}\, x4^{7}\, x6^{4}\, x8^{4}\, x9\, x12^{3}\, x14^{11}\, x15^{3}\, x101^{3}\, x102^{2}\, x103^{2}\, x106^{7} \\
x82^{4}\, x84^{2}\, x87\, x90^{8}\, x91^{9}\, x93\, x97\, x98^{6}\, x99^{3}\, x100^{4}\, x43^{2}\, x51\, x52^{3}\, x53\, x55\, x58\, x60^{3}\, x61^{8}\, x62^{2}\, x63^{14}\, x64^{2}\, x65^{2}\, x66^{6}\, x71^{2}\, x72^{7}\, x74^{6}\, x75\, x76^{5}\, x77\, x79^{3}\, x81 \\
-x109^{5}\, x110^{4}\, x111\, x112\, x113^{8}\, x114^{6}\, x115\, x116^{3}\, x117^{2}\, x118^{2}\, x119^{2}\, x120^{4}\, x121\, x122\, x124^{2}\, x125^{3}\, x126^{4}\, x127^{5}\, x128^{7}\, x129^{5}\, x131\, x132^{10}\, x133\, x135^{3}\, x140^{2} \\
x142\, x144\, x146^{3}\, x147^{9}\, x148^{5}\, x194\, x195^{2}\, x196^{3}\, x197^{2}\, x198^{5}\, x199^{4}\, x200\, x150^{6}\, x151^{2}\, x153^{6}\, x155^{6}\, x156^{4}\, x157^{4}\, x158^{6}\, x160^{2}\, x161^{11}\, x162^{5}\, x163^{8}\, x164^{5}\, x166^{4} \\
x168^{2}\, x169\, x171\, x172^{5}\, x173\, x174^{10}\, x176^{11}\, x177\, x178^{2}\, x179^{2}\, x180\, x181\, x182^{7}\, x183^{2}\, x185^{4}\, x186\, x187^{3}\, x188^{3}\, x189\, x190^{3}\, x192\, x16^{14}\, x17^{4}\, x18^{10}\, x20^{2}\, x21^{6} \\
x23\, x25\, x26\, x27^{2}\, x28^{2}\, x29\, x31^{2}\, x32^{8}\, x33^{5}\, x34\, x36\, x37^{8}\, x38\, x39\, x41^{2}\, x42^{2}\, x44\, x45\, x46^{3}\, x47\, x48^{3}\, x49^{4}\, x2\, x3\, x4^{5}\, x5\, x7^{2}\, x8^{7}\, x9^{2}\, x10^{2}\, x11^{2}\, x12^{4}\, x13^{2}\, x14^{11}\, x15 \\
x102\, x103\, x104\, x106\, x82^{2}\, x84^{3}\, x85^{3}\, x86\, x87\, x88^{5}\, x90^{9}\, x91\, x92^{2}\, x93^{2}\, x95\, x96^{8}\, x97^{4}\, x98\, x99^{4}\, x100^{7}\, x43^{6}\, x51^{2}\, x52^{3}\, x54^{2}\, x55\, x63^{9}\, x64\, x65\, x66^{5}\, x68\, x70^{2}\, x72^{2} \\
x74\, x75^{2}\, x76^{2}\, x77^{5}\, x78^{2}\, x79^{2}\, x80\, x81^{10},\ [x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17, x18, x19, x20, x21, x22, x23, x24, x25, x26, \\
x27, x28, x29, x30, x31, x32, x33, x34, x35, x36, x37, x38, x39, x40, x41, x42, x43, x44, x45, x46, x47, x48, x49, x50, x51, x52, x53, x54, x55, x56, x57, x58, \\
x59, x60, x61, x62, x63, x64, x65, x66, x67, x68, x69, x70, x71, x72, x73, x74, x75, x76, x77, x78, x79, x80, x81, x82, x83, x84, x85, x86, x87, x88, x89, x90, \\
x91, x92, x93, x94, x95, x96, x97, x98, x99, x100, x101, x102, x103, x104, x105, x106, x107, x108, x109, x110, x111, x112, x113, x114, x115, x116, x117, \\
x118, x119, x120, x121, x122, x123, x124, x125, x126, x127, x128, x129, x130, x131, x132, x133, x134, x135, x136, x137, x138, x139, x140, x141, x142, \\
x143, x144, x145, x146, x147, x148, x149, x150, x151, x152, x153, x154, x155, x156, x157, x158, x159, x160, x161, x162, x163, x164, x165, x166, x167, \\
x168, x169, x170, x171, x172, x173, x174, x175, x176, x177, x178, x179, x180, x181, x182, x183, x184, x185, x186, x187, x188, x189, x190, x191, x192, \\
x193, x194, x195, x196, x197, x198, x199, x200]
\end{gathered}
\tag{21}
$$

```
>

> HIGH_DEGREE_POLYNOMIAL_INTERPOLATION(P,vars);
```

```
Evaluations time=    0.0020s

   Solve time=    0.0010s
   Roots time=    0.0040s
 ModularLog time=    0.0610s
Evaluations time=    0.0020s
   Solve time=    0.0010s
   Roots time=    0.0030s
 ModularLog time=    0.0280s
Evaluations time=    0.0020s
   Solve time=    0.0020s
   Roots time=    0.0030s
 ModularLog time=    0.0360s
Evaluations time=    0.0020s
   Solve time=    0.0010s
   Roots time=    0.0080s
 ModularLog time=    0.0310s
Evaluations time=    0.0020s
   Solve time=    0.0010s
   Roots time=    0.0040s

 ModularLog time=    0.0350s
Evaluations time=    0.0020s
   Solve time=    0.0020s
   Roots time=    0.0040s
 ModularLog time=    0.0300s
Evaluations time=    0.0020s
   Solve time=    0.0010s
   Roots time=    0.0050s
 ModularLog time=    0.0270s
Evaluations time=    0.0020s
   Solve time=    0.0010s
   Roots time=    0.0040s
 ModularLog time=    0.0260s
Evaluations time=    0.0020s
   Solve time=    0.0010s
   Roots time=    0.0040s

 ModularLog time=    0.0970s
Evaluations time=    0.0030s
   Solve time=    0.0010s
   Roots time=    0.0050s
 ModularLog time=    0.0320s
Evaluations time=    0.0020s
   Solve time=    0.0010s
   Roots time=    0.0040s
```

```
ModularLog time=    0.0290s
Evaluations time=    0.0020s
   Solve time=    0.0020s
   Roots time=    0.0050s
ModularLog time=    0.0250s
Evaluations time=    0.0020s
   Solve time=    0.0020s
   Roots time=    0.0060s
ModularLog time=    0.0250s
```

"constructed Polynomial = original polynomial: " *true* (22)

>

>

>

>

>

>