Powers *permet de calculer a^k mod n en décomposant k en base 2.*

```
> Powers := proc(n, a, k)
  local X, Z, r, Y, A, i, j;
  X := convert(k, 'base', 2);   (*Convertir k en base 2*)
  Z := convert(X, array);
  r := 0;   Y := 1;
  while 2^r <= k do r := r + 1  (* r est le nombre de chiffre de X en base 2*) end do;
  A := Matrix(1, r); A[1, 1] := a;
  for i from 2 to r do A[1, i] := A[1, i - 1]^2 mod n end do;  (*relation de récurrence *)
  for j to r do if Z[j] = 1 then Y := Y* A[1, j] end if end do; Y mod n end proc
```

Algorithme p-1 de Pollard

```
> Pollard := proc(n)
  local a, d, b, g;
   a := 3;
   for d from 2 to 1000! do
   b := Powers(n, a, d);
   g := gcd(b - 1, n);
   if (g > 1) then
  return(g);
   end if;
   end do;
   end proc;
> Pollard(1715761513)
```

26927 **(1)**

>

double *permet de calculer kP avec P=(x1,y1) et la courbe elliptique donnée par y^2=x^3+bx+c*

```
> double := proc(k, x1, y1, b, c, n)
   local r, X, Z, A, m, λ, ν, i, x, y, Y, x3, y3, j, l;
  X := convert(k, 'base', 2);
  Z := convert(X, array);
   r := 0;
  while 2^r <= k do r := r + 1 end do;
   A := Matrix(1, 2 r);
        (*A est un vecteur où A=(x(P),y(P),x(2P),y(2P),x(4P),y(4P),x(8P),y(8P),....)*)
   A[1, 1] := x1 ;
  A[1, 2] := y1 ;
   for j from 1 to r - 1 do
   if gcd(2· A[1, 2 j], n) = 1 and gcd(4·A[1, 2 j - 1]^3 + 4·b·A[1, 2 j - 1] + 4 c, n) = 1 then
```

$$A[1,2j+1] := \frac{\left(A[1,2j-1]^4 - 2\,b\cdot A[1,2j-1]^2 - 8\cdot c\cdot A[1,2j-1] + b^2\right)}{4\cdot A[1,2j-1]^3 + 4\cdot b\cdot A[1,2j-1] + 4\,c} \ \mathbf{mod}\ n;$$

(*Formule de duplication, coordonnée en x*)

```
λ := (3·A[1, 2j − 1]^2 + b) / (2·A[1, 2j]) mod n;
 ν := A[1, 2j] − λ·A[1, 2j − 1];
 A[1, 2j + 2] := -(λ·A[1, 2j + 1] + ν) mod n ; (*Formule de duplication, coordonnée en y*)
 else return max(gcd(A[1, 2j], n), gcd(4·A[1, 2j − 1]^3 + 4·b·A[1, 2j − 1] + 4 c, n));
 end if;
 end do;
 m := 1;
for i from 1 to r  do
 if Z[i] = 0 then
 m := m + 1;
 end if;
 end do;
 x := A[1, 2m − 1];
 y := A[1, 2m];
 Z[m] := 0;
 for l  from 1 to  r  do  (*Somme des P(l) tels que k(l)=1*)
if (Z[l] = 1) then
 X := A[1, 2l − 1]; Y := A[1, 2l];
 if (gcd(X − x, n) = 1) then
  λ := (Y − y) / (X − x) mod n;
 x3 := (λ^2 − x − X);
 y3 := ( -λ·x3 − (y − λ·x));
 x := x3 mod n;
 y := y3 mod n;
 elif ( gcd(X − x, n) < n) then return gcd(X − x, n);
 elif  (gcd(X − x, n) = n) then print'("refaire");
 end if;
 end if;
 end do;
 [x, y];
 end proc;
```

(2)

Factorisation de Lenstra par les courbes elliptiques ou ECM

```
> Lenstra :=proc(n)
 local c, Q, p, b, x1, y1
 if n :: prime  then
 return(print("n est premier"));
 end if;
 b := rand( ); x1 := rand( ); y1 := rand( );
  c := (y1^2 − x1^3 − b·x1 ) mod n;
 if (gcd(4·b^3 + 27·c^2, n) > 1) then
 return  gcd(4·b^3 + 27·c^2, n);
 end if ;
 for p from 2 to 1000! do    (* p est borné par un nombre que pourait choisir l'utilisateur *)
 Q := double(p!, x1, y1, b, c, n);
```

```
if (Q :: integer) then
return Q;  (* Si on est dans ce cas,  on a trouvé un facteur de n *)
end if;
b := rand( ); x1 := rand( ); y1 := rand( );
c := (y1^2 - x1^3 - b·x1) mod n;
if (gcd(4·b^3 + 27·c^2, n) > 1) then
return gcd(4·b^3 + 27·c^2, n);
end if ;
end do;
print("Réessayer avec une nouvelle courbe et/ou point");
Q;
end proc;
```

Exemple du livre de Silverman Section 4.4

```
> Lenstra(1715761513)
```

$$26927 \quad (3)$$

```
> double(5, 2, 3, 1, 0, 17)
```

$$[7, 10] \quad (4)$$

```
>
>
> echelle(37, 8, 5, 4, 1715761513)
```

$$\begin{bmatrix} 1364550293 & 1527563677 \end{bmatrix} \quad (5)$$

```
> echelle :=proc(k, x1, a, b, n)
local r, m, h, X, Z, Q, P, j;
r := 0;
m := Matrix(1, 2 ); h := Matrix(1, 2 );
while 2^r <= k do r := r + 1 end do;
X := convert(k, 'base', 2);
Z := convert(X, array);
Q := Matrix(1, 2 ); P := Matrix(1, 2 );
Q[1, 1] := x1 mod n;
Q[1, 2] := 1 ;
P[1, 1] := (x1 + 1)^2·(x1 - 1)^2 mod n;
P[1, 2] := (4·x1)((x1)^2 + (a + 2)/4 ·4·x1) mod n;
for j from 0 to r - 2 do
if Z[(r - 1) - j] = 1 then
if gcd(Q[1, 1], n) = 1 and gcd(Q[1, 2], n) = 1 and gcd(P[1, 1], n) = 1 and gcd(P[1, 2], n)
    = 1 then
h[1, 1] := ((P[1, 1] - P[1, 2])(Q[1, 1] + Q[1, 2]) + (P[1, 1] + P[1, 2])(Q[1, 1]-Q[1,
```

```
    2]))^2 mod n;
h[1, 2] := x1·((P[1, 1] − P[1, 2])(Q[1, 1] + Q[1, 2]) - (P[1, 1] + P[1, 2])(Q[1, 1]-Q[1,
    2]))^2 mod n;
m[1, 1] := (P[1, 2] + P[1, 1])^2·(P[1, 1] − P[1, 2])^2 mod n;
m[1, 2] := (4·P[1, 1])((P[1, 1])^2 + (a+2)/4·4·P[1, 1]·P[1, 2]) mod n;
Q[1, 1] := h[1, 1];
Q[1, 2] := h[1, 2];
P[1, 1] := m[1, 1];
P[1, 2] := m[1, 2];
else return(max(gcd(Q[1, 1], n), gcd(Q[1, 2], n), gcd(P[1, 1], n), gcd(P[1, 2], n)));
end if;
else
if gcd(Q[1, 1], n) = 1 and gcd(Q[1, 2], n) = 1 and gcd(P[1, 1], n) = 1 and gcd(P[1, 2], n)
    = 1 then
m[1, 1] := ((P[1, 1] − P[1, 2])(Q[1, 1] + Q[1, 2]) + (P[1, 1] + P[1, 2])(Q[1, 1]-Q[1,
    2]))^2 mod n;
m[1, 2] := x1·((P[1, 1] − P[1, 2])(Q[1, 1] + Q[1, 2]) - (P[1, 1] + P[1, 2])(Q[1, 1]-Q[1,
    2]))^2 mod n;
h[1, 1] := (Q[1, 2] + Q[1, 1])^2·(Q[1, 1] − Q[1, 2])^2 mod n;
h[1, 2] := (4·Q[1, 1])((Q[1, 1])^2 + (a+2)/4·4·Q[1, 1]·Q[1, 2]) mod n;
Q[1, 1] := h[1, 1];
Q[1, 2] := h[1, 2];
P[1, 1] := m[1, 1];
P[1, 2] := m[1, 2];
else return(max(gcd(Q[1, 1], n), gcd(Q[1, 2], n), gcd(P[1, 1], n), gcd(P[1, 2], n)));
end if;
end if;
end do;
return(Q);
end proc;
```

```
> echelle(3, 45, 6, 5, 1715761513)
```

$$\begin{bmatrix} 286260492 & 5832000 \end{bmatrix} \quad (6)$$

```
> Lenstra2 :=proc(n)
local b, Q, p, a, x1, y1;
a := rand( );
x1 := rand( );
for p from 2 to 300000000 do    (* p est borné par un nombre que pourait choisir l'utilisateur *)
Q := echelle(p, x1, a, b, n); a := rand( );
x1 := rand( );
if (Q :: integer) then
return Q;  (* Si on est dans ce cas,  on a trouvé un facteur de n *)
end if;
end do;
print("Réessayer avec une nouvelle courbe et/ou point");
Q;
end proc;
```

```
> with(numtheory)
> card := proc(a, b, c, p)
      local x, y, s;
      s := 0;
      for x from 0 to p-1 do
         y := x^3 + a·x^2 + b·x + c;
         s := s + 1 + legendre(y, p);
      od;
      return (s + 1);
  end;
```

```
> card(1525453, 45646, 166, 11)
```

$$15 \tag{7}$$

```
>
>
>
>
>
```