```
> restart;
```

Restes chinois

```
>
```

Restes chionois pour les entiers :

```
> chrem( [1, 2, 3], [5, 7, 11] );
```

$$366 \tag{1.1}$$

```
> 366 mod 5; 366 mod 7; 366 mod 11;
```

$$1$$
$$2$$
$$3 \tag{1.2}$$

```
>
```

Ca marche.

J'écris maintenant un programme des restes chinois pour les polynômes.
Ce programme rend le polynôme de plus petit degré P congru à
B[i]) mod M[i] pour tout i.

```
> chrempol := proc(B, M)
  local m, n, a, i, c;
  n := nops(B);
  m := product(M[i], i = 1..n);
  for i from 1 to n do
      a[i] := m/M[i];
      gcdex(a[i], M[i], x,'s','t');
      c[i] := rem(s*B[i], M[i], x);
  od;
  expand(sum(expand(c[j]*a[j]), j = 1..n));
  end;
```

```
chrempol := proc(B, M)                                  (1.3)
  local m, n, a, i, c;
  n := nops(B);
  m := product(M[i], i = 1..n);
  for i to n do
    a[i] := m / M[i];
    gcdex(a[i], M[i], x, 's', 't');
    c[i] := rem(s*B[i], M[i], x)
  end do;
  expand(sum(expand(c[j]*a[j]), j = 1..n))
end proc
```

```
> B := [1, 2, 3, 4]; M := [x, x-3, x+1, x+3];
```

$$B := [1, 2, 3, 4]$$
$$M := [x, x-3, x+1, x+3] \quad (1.4)$$

>

Normalement, on devrait obtenir le polynôme P de degré inférieur ou égal à 3 tel que P(0)=1, P(3)=2, P(-1)=3 et P(-3)=4.

> $P := chrempol(B, M);$

$$P := \frac{13}{72}x^3 + \frac{2}{9}x^2 - \frac{47}{24}x + 1 \quad (1.5)$$

> $P := unapply(P, x);$

$$P := x \rightarrow \frac{13}{72}x^3 + \frac{2}{9}x^2 - \frac{47}{24}x + 1 \quad (1.6)$$

> $P(0); P(3); P(-1); P(-3);$

$$1$$
$$2$$
$$3$$
$$4 \quad (1.7)$$

>

On doit trouver la même chose en utilisant la fonction "interp".

> $interp([0, 3, -1, -3], [1, 2, 3, 4], x);$

$$\frac{13}{72}x^3 + \frac{2}{9}x^2 - \frac{47}{24}x + 1 \quad (1.8)$$

>

>

Mais on peut appliquer la fonction chrempol à des polynômes de degré supérieur strictement à 1 (pour M). Ce n'est pas le cas pour la fonction "interp" de maple.

> $B := [x^2 + 1, x - 1, 2]; M := [x^3 + 1, x^4, x + 2];$

$$B := [x^2 + 1, x - 1, 2]$$
$$M := [x^3 + 1, x^4, x + 2] \quad (1.9)$$

>

Les M[i] sont deux à deux premiers entre eux :

> $gcd(M[1], M[2]); gcd(M[2], M[3]); gcd(M[1], M[3]);$

$$1$$
$$1$$
$$1 \quad (1.10)$$

> $P := chrempol(B, M);$

$$P := \frac{171}{112}x^7 + 2x^6 - x^5 + \frac{283}{112}x^4 + x - 1 \quad (1.11)$$

> $for\ i\ from\ 1\ to\ 3\ do\ rem(P, M[i], x)\ od;$

$$x^2+1$$
$$x-1$$
$$2 \qquad (1.12)$$

>

Générateur de (Z/pZ)*

>

Ici, on considère un nombre premier p impair. Pour trouver un générateur de (Z/pZ)*, il y a la commande **primroot**, qui marche en fait pour tout entier n tel que (Z/nZ)* est cyclique.

On va quand même programmer une telle recherche, à titre d'exercice.

```
> generateur:= proc(p, decomp)
  local r, a, b, c, n, i;
  r:= rand(2..p-1);
  n:= nops(decomp);
  for i from 1 to n do
          b:= 1;
          while b=1 do
                a:= r();
                b:= a^((p-1))/(decomp[i][1]) mod p;
          od;
          c[i]:= a^((p-1))/(decomp[i][1]decomp[i][2]) mod p;
  od;
  product(c[j], j=1..n) mod p;
  end;
```

$$generateur := \mathbf{proc}(p, decomp) \qquad (2.1)$$

```
local r, a, b, c, n, i;
r:= rand(2..p-1);
n:= nops(decomp);
for i to n do
    b:= 1;
    while b=1 do
        a:= r();
        b:= mod(a^((p-1)/(decomp[i])[1]), p)
    end do;
    c[i]:= mod(a^((p-1)/(decomp[i])[1]^(decomp[i])[2]), p)
end do;
mod(product(c[j], j=1..n), p)
```

```
end proc
> ifactors(6);
```

$$[1, [[2, 1], [3, 1]]] \tag{2.2}$$

```
> decomp:=% [2];
```

$$decomp := [[2, 1], [3, 1]] \tag{2.3}$$

```
> generateur(7, decomp);
```

$$5 \tag{2.4}$$

On peut vérifier l'ordre.

```
> with(numtheory);
Warning, the protected name order has been redefined and
unprotected
```

$$[GIgcd, bigomega, cfrac, cfracpol, cyclotomic, divisors, factorEQ, factorset, fermat, imagunit, index, integral_basis, invcfrac, invphi, issqrfree, jacobi, kronecker, \lambda, legendre, mcombine, mersenne, migcdex, minkowski, mipolys, mlog, mobius, mroot, msqrt, nearestp, nthconver, nthdenom, nthnumer, nthpow, order, pdexpand, \phi, \pi, pprimroot, primroot, quadres, rootsunity, safeprime, \sigma, sq2factor, sum2sqr, \tau, thue] \tag{2.5}$$

```
> order(5, 7);
```

$$6 \tag{2.6}$$

```
> hasard:= rand(1..1000000);
hasard := proc()
    (proc()
        option builtin;
        391
    end proc)(6, 1000000, 20) + 1
end proc
```

(2.7)

```
> p:= nextprime(hasard());
```

$$p := 144701 \tag{2.8}$$

```
> decomp:= ifactors(p- 1)[2];
```

$$decomp := [[2, 2], [5, 2], [1447, 1]] \tag{2.9}$$

```
> generateur(p, decomp);
```

$$46843 \tag{2.10}$$

```
> order(%, p);
```

$$144700 \tag{2.11}$$

```
>
```

Générateur de (Z/p^nZ)*

```
> p:='p';
```

$$p := p \tag{3.1}$$

Soit p un nombre premier impair, et soit n un entier supérieur strictement à 1. Alors (Z/p^nZ)* est cyclique.
De plus, si g est une racine primitive modulo p, alors si ce n'est pas une racine primitive modulo p^n, alors g+p en est une.

De plus, pour voir si g est une racine primitive modulo p^n, il suffit de voir si

```
> g^((p-1)p^(n-2)) mod p^n <> 1;
```

$$modp\left(g^{((p-1)p^{(n-2)})}, p^n\right) \neq 1 \tag{3.2}$$

```
>
```

Là encore, on pourrait utiliser **primrroot**. On fait ce programme à titre d'exercice.

```
> primroot(7^45);
```

$$3 \tag{3.3}$$

```
> order(3, 7^45);
```

$$91720203791655457162591114615660958406 \tag{3.4}$$

```
> phi(7^45);
```

$$91720203791655457162591114615660958406 \tag{3.5}$$

```
> gene2 := proc(p, n, decomp)
  local g;
  g := generateur(p, decomp);
  if g^((p-1)*p^(n-2)) mod p^n = 1 then g+p else g fi;
  end;
```

```
gene2 := proc(p, n, decomp)                                    (3.6)
    local g;
    g := generateur(p, decomp);
    if mod(g^((p-1)*p^(n-2)), p^n) = 1 then
        g + p
    else
        g
    end if
end proc
```

```
> decomp := ifactors(6)[2];
```

$$decomp := [[2, 1], [3, 1]] \tag{3.7}$$

```
> debug(gene2); gene2(7, 45, decomp);
```

```
                                gene2
{--> enter gene2, args = 7, 45, [[2, 1], [3, 1]]
                                g := 5
<-- ERROR in gene2 (now at top level) = numeric exception:
overflow}
Error, (in gene2) numeric exception: overflow
```

```
> 5^(6·7^43) mod 7^45;
```
Error, numeric exception: overflow (3.8)

$$5^{6\cdot 7^{43}} \bmod 7^{45};$$

```
> 5&^(6*7^43) mod 7^45;
```
$$917202037916554571625911146156609584 07 \quad (3.9)$$

>

OK. J'ai des problèmes avec cette interface. Je réécris le programme, en mode Text.

```
> gene2:=proc(p,n,decomp)
  local g;
  g:=generateur(p,decomp);
  if g&^((p-1)*p^(n-2)) mod p=1 then g+p else g fi;
  end;
```

```
gene2 := proc(p, n, decomp)
    local g;
    g := generateur(p, decomp);
    if mod(g &^ ((p - 1) * p^(n - 2)), p) = 1 then
        g + p
    else
        g
    end if
end proc
```
(3.10)

```
> gene2(7, 45, decomp);
```
$$10 \quad (3.11)$$

```
> order(10, 7^45);
```
$$917202037916554571625911146156609584 06 \quad (3.12)$$

```
> phi(7^45);
```
$$917202037916554571625911146156609584 06 \quad (3.13)$$

```
> p := ithprime(34567);
```
$$p := 409463 \quad (3.14)$$

```
> decomp := ifactors(p-1)[2];
```
$$decomp := [[2, 1], [17, 1], [12043, 1]] \quad (3.15)$$

```
> g := gene2(p, 100, decomp);
```
$$g := 428546 \quad (3.16)$$

```
> order(%, p^100);
```
$$1665195940328335132346691270870056364362795955353901098731 91\ \quad (3.17)$$
$$0956570862546320375835485758428126857026113578057367513\$$
$$5810733430634736186167873752267791655417139042236288338\$$

```
      047979116818380971612144935992661130596780115076755975 8\
      200492286702440547062726369800996022910193286272114252 3\
      462890360213212414351737384105893672574430181797288874 2\
      741719716065825999919664674080990967211210987467899482 0\
      632998614934277682778058417738847540063961344550978254\
      845734946990123031264560627853831223842607927568960741 9\
      818853323103806413239551247650585176657944574011953032 9\
      4164474
> phi(p^100);
166519594032833513234669127087005636436279595535390109873191\
      095657086254632037583548575842812685702611357805736751 3\
      581073343063473618616787375226779165541713904223628833 8\
      047979116818380971612144935992661130596780115076755975 8\
      200492286702440547062726369800996022910193286272114252 3\
      462890360213212414351737384105893672574430181797288874 2\
      741719716065825999919664674080990967211210987467899482 0\
      632998614934277682778058417738847540063961344550978254\
      845734946990123031264560627853831223842607927568960741 9\
      818853323103806413239551247650585176657944574011953032 9\
      4164474
>
```

(3.18)

Pour vérifier que g est un générateur, on peut aussi faire comme suit.

```
> estgene:=proc(g,p,n,decomp)
  local r,i,k;
  r:=1;
  k:=nops(decomp);
  for i from 1 to k do
      if g&^((p-1)/decomp[i][1]) mod p=1 then return(0) fi;
  od;
  if g&^((p-1)*p^(n-2)) mod p^n=1 then r:=0 fi;
  r;
  end;
```

$$estgene := \mathbf{proc}(g, p, n, decomp) \tag{3.19}$$

```
    local r, i, k;
    r := 1;
    k := nops(decomp);
    for i to k do
        if mod(g &^ ((p - 1) / (decomp[i])[1]), p) = 1 then
            return 0
```

```
        end if
    end do;
    if mod(g &^ ((p - 1) * p^(n - 2)), p^n) = 1 then
        r := 0
    end if;
    r
end proc
```

```
> estgene(g, p, 100, decomp);
```

$$1 \tag{3.20}$$

C'est un générateur.

```
> estgene(2, p, 100, decomp);
```

$$0 \tag{3.21}$$

Ce n'est pas un générateur.

```
> order(2, p^100);
```

$$\begin{aligned}&832597970164167566173345635435028182181397977676950549365955\backslash\\&4782854312731601879177428792140634285130567890286837567\backslash\\&9053667153173680930839368761338958277085695211181441690\backslash\\&2398955840919048580607246799633056529839005753837798791\backslash\\&0024614335122027353136318490049801145509664313605712617\backslash\\&3144518010660620717586869205294683628721509089864443713\backslash\\&7085985803291299995983233704049548360560549373394974103\backslash\\&1649930746713888413890292088694237700319806722754891274\backslash\\&2286747349506151563228031392691561192130396378448037099\backslash\\&0942666155190320661977562382529258832897228700597651647\backslash\\&082237\end{aligned} \tag{3.22}$$

```
>
```