```
######################## PROBLÈME 1: ########################
# 1) solution itérative
pgcd := proc(A,B)
  local t,a,b ;
  a := A ; b := B ; # Maple refuse de modifier les paramètres
  if (a = 0) then RETURN (b) ; fi ;
  while (b <> 0) do
    t := b ;
    b := a mod b ;
    a := t ;
  od ;
  a ;
end:

# solution récursive
pgcd2 := proc(a,b)
  if (b = 0) then  a ;
  else             pgcd2(b, a mod b) ;
  fi ;
end:

# 2)
Bezout := proc(a,b)
  local t,q,r,x,y,vx,vy ;
  if (a = 0) then RETURN ([b,0,1]) ; fi ;

  x := a ; y := b ; vx := 0 ; vy := 1 ;
  while (y <> 0) do
    q := iquo(x,y, 'r') ;
    t := vy ;
    vy := vx - q*vy ;
    vx := t ;

    x := y ;
    y := r ;
  od ;
  [x, (x-b*vx)/a, vx] ;
end:

# 3), 5)
with(linalg):

# renvoie l'indice de l'élément non nul de valeur absolue minimial
IndMin := proc(L)
  local i, k, m ;
  k := 1 ; m := abs(L[k]) ;
  for i from 2 to nops(L) do
```

```
    if (L[i] <> 0 and abs(L[i]) < m) then
      k := i ; m := abs(L[k]) ;
    fi
  od ;
  k
end:
# A0 liste, renvoie B = (delta,0...0) et U tq A U = B
BezoutN := proc(A0)
  local n, U, A, i, k, m, q, r ;
  A := A0 ; n := nops(A) ;
  U := array(identity, 1..n,1..n) ;
  k := IndMin(A) ; m := abs(A[k]) ;
  while true do
    print(A, k) ; # pour le voir fonctionner...
    if (k > 1) then
      i := A[1] ; A[1] := A[k] ; A[k] := i ;
      U := swapcol(U, 1, k) ;
    fi ;
    for i from 2 to n do
      q := iquo(A[i], m, 'A[i]') ;
      U := addcol(U, 1, i, -q) ;
    od ;
    k := IndMin(A) ; m := abs(A[k]) ;
    if (k = 1) then RETURN ([A,U]) ; fi
  od ;
end:
# Exemple:
A0 := [116085838, 181081878, 314252913, 10346840]:
#
# comparer BezoutN(A0) avec ihermite(matrix(4,1,A0), 'U')
# La plus petite solution (norme L2) est [-88, 352, -167, -101]
A1 := [763836, 1066557, 113192, 1785102, 1470060, 3077752,
       114793, 3126753, 1997137, 2603018]:
# La plus petite solution est [3, -1, 1, 2, -1, -2, -2, -2, 2, 2]
######################## PROBLÈME 2: ########################
# 1)
InvMod := proc(x, N)
  local D,d,u ;
  # on peut remplacer les deux lignes suivantes par
  # d := igcdex(x, N, 'u') ;
  D := Bezout(x,N) ;
  d := D[1] ; u := D[2] ;
```

```
  if (d <> 1) then
    false ;
  else
    u mod N ;
  fi
end:
# bien sûr, 1/x mod N marche directement!
# 2) résoud x = A mod a, x = B mod b [on suppose ab <> 0]
Chinois := proc(A,B, a, b)
  local D,d,u,v,m,x,M ;
  D := Bezout(a,b) ;
  d := D[1] ; u := D[2] ; v := D[3] ;
  if (d <> 1 and A mod d <> B mod d) then
    false ;
  else
    x := A + u*(a/d)*(B-A) ;
    M := a*b/d ;
    [x mod M, M]
  fi ;
end:
# noter que chrem ne sait pas traiter le cas (a,b) > 1
# 3) x = liste de résidus, m = liste de modules, de même longueurs
ChinoisN := proc(x, m)
  local i,d,X,M ;
  X := x[1] ; M := m[1] ;
  for i from 2 to nops(x) do
    d := Chinois(X,x[i], M,m[i]) ;
    if (d = false) then RETURN(false) ; fi ;
    X := d[1] ; M := d[2] ;   # X mod M solution des congruences 1..i
  od ;
  [X, M] ;
end:
# une solution "diviser pour régner" serait plus efficace
# 4)
ChinoisN([3,2,5,5], [14,13,11,8]) ;
######################## PROBLÈME 3: ########################
# 1) on suppose n > 0. Renvoie (e_0, ..., e_k)
chiffres := proc(n0, B)
  local l,n ;
  n := n0 ; l := NULL ;
  while (n > 0) do
    l := l, irem(n,B,'n') ;
```

```
  od ;
  [l] ;
end:
bits := n -> chiffres(n,2) ;
# 2) ne marche pas si n = 0
Pow := proc(x, n)
  local i,y,z, e,k ;
  y := 1 ; z := x ;
  e := bits(n) ; k := nops(e) - 1 ;
  for i from 0 to k do
    print (i,y,z) ; # pour le voir fonctionner
    if (e[i+1] = 1) then y := y*z ; fi ;
    z := z^2 ; # inutile si i = k
  od ;
  y ;
end:
Pow(x,10) ;
# plus simple, marche aussi si n = 0
PowDG := proc(x, n0)
  local i,n,y,z ;
  y := 1 ; z := x ; n := n0 ;
  while (n > 0) do
    if (irem(n, 2, 'n') = 1) then y := y*z ; fi ;
    if (n = 0) then RETURN (y) ; fi ; # évite une mise au carré inutile
    z := z^2 ;
  od ;
  y ;
end:
# 3) on lit les bits a l'envers
PowGD := proc(x, n)
  local i,y, e,k ;
  if (n = 0) then RETURN (1) ; fi ;
  y := x ;
  e := bits(n) ; k := nops(e) - 1 ;
  for i from k to 1 by -1 do
    y := y^2 ;
    if (e[i] = 1) then y := y*x ; fi ;
  od ;
  y ;
end:
```