```
    |\^/|     Maple 9 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2003
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> interface(screenwidth=120);
#non on n'a pas utilis\'e le fait que b soit reduite, c'est l'in\'ealit\'e d'Hadamard
#C'est deja programme dans maple: l'instruction lattice existe deja
#dans maple7 et superieur. Dans maple9 il y a en plus le paquet with(IntegerRelations);
#qui contient l'instruction LLL.
#Pour xcas il y a l'instruction: lll
> with(LinearAlgebra):
> M:=Matrix([[1,2,3],[-1,0,1],[0,1,1]]);
                                                    [ 1    2    3]
                                                    [             ]
                                                M := [-1    0    1]
                                                    [             ]
                                                    [ 0    1    1]

# lattice attend une liste et non une Matrix ni une matrix
> lattice(convert(M,listlist));
                                          [[-1, 0, 1], [0, 1, 1], [0, -1, 1]]

> #On teste d'abord l'orthogonalisation
> ortho:=proc(A);
> b:=convert(Transpose(A),listlist);
> #la liste des vecteurs b, bb son orthogonalise.
> b:=[seq(Vector(i),i=b)];
> n:=nops(b);#la dimension du reseau
> bb:=b;
> for i from 2 to n do
> bb[i]:=simplify(b[i]-add((b[i].bb[j])/(bb[j].bb[j])*bb[j],j=1..i-1));
> od;
> mu:=Matrix(n,n,(i,j)->(b[i].bb[j])/(bb[j].bb[j]));
> mu,bb;
> end proc;
ortho := proc(A)
local b, n, bb, i, mu;
    b := convert(LinearAlgebra:-Transpose(A), listlist);
    b := [seq(Vector(i), i = b)];
    n := nops(b);
    bb := b;
    for i from 2 to n do bb[i] := simplify(b[i] - add(((b[i]) . (bb[j]))*bb[j]/(bb[j]) . (bb[j]), j = 1 .. i - 1))
    end do;
    mu := Matrix(n, n, (i, j) -> (b[i]) . (bb[j])/(bb[j]) . (bb[j]));
    mu, bb
end proc

> v:=ortho(M)[2];v[1].v[2],v[2].v[3],v[3].v[1];
                                                  [ 1]  [1]  [1/3 ]
                                                  [  ]  [ ]  [    ]
                                              v := [[-1], [1], [1/3 ]]
                                                  [  ]  [ ]  [    ]
                                                  [ 0]  [1]  [-2/3]

                                                    0, 0, 0

> ########################################
> myLLL:=proc(A);
> #Attention pour convertir en des Vecteurs par cette intruction il faut transposer
> b:=convert(Transpose(A),listlist);
> #la liste des vecteurs b, bb son orthogonalise.
> b:=[seq(Vector(i),i=b)];
> n:=nops(b);#la dimension du reseau
> bb:=b;
> for i from 2 to n do
> bb[i]:=simplify(b[i]-add((b[i].bb[j])/(bb[j].bb[j])*bb[j],j=1..i-1));
> od;
> mu:=Matrix(n,n,(i,j)->(b[i].bb[j])/(bb[j].bb[j]));
> #BB= les carre des normes des b^*
> BB:=[seq(bb[i].bb[i],i=1..n)];
> k:=2;
> #
> while ( k<n+1)
> do
> if abs(mu[k,k-1])> 0.5 then
>        r:=round(mu[k,k-1]);
>        b[k]:=b[k]-r*b[k-1];
>        for j from 1 to k-2 do mu[k,j]:=mu[k,j]-r*mu[k-1,j];od;
>        mu[k,k-1]:=mu[k,k-1]-r;
> fi;
> if ( BB[k] < BB[k-1]*(3/4-mu[k,k-1]^2))
> then
> mk:=mu[k,k-1];#on sauve l'ancienne valeur de mu[k,k-1]
>  #cc=norme de c^*_{k-1} au carre
>  cc:=BB[k]+mk^2*BB[k-1];
> #on remplace mu[k,k-1] par la valeur de nu_{k,k-1}
> mu[k,k-1]:=mk*BB[k-1]/cc;
> for i from k+1 to n do
> mik:=mu[i,k-1];#on sauvegarde
> mu[i,k-1]:=mu[i,k-1]*mu[k,k-1]+mu[i,k]*BB[k]/cc;
> mu[i,k]:=mik-mu[i,k]*mk;
> od;
> for j from 1 to k-2 do
> mkj:=mu[k,j];mu[k,j]:=mu[k-1,j];mu[k-1,j]:=mkj;
> od;
> #on echange b_k et b_{k-1}
> c:=b[k];b[k]:=b[k-1];b[k-1]:=c;
> #on corrige le carre des normes de b^*
> BB[k]:=BB[k-1]*BB[k]/cc;BB[k-1]:=cc;
> k:=max(2,k-1);
> else
> for l from k-2 by -1 to 1
>    do
>    if abs(mu[k,l])>0.5 then
>    r:=round(mu[k,l]);b[k]:=b[k]-r*b[l];
>    for j from 1 to l-1 do mu[k,j]:=mu[k,j]-r*mu[l,j] od;
>    mu[k,l]:=mu[k,l]-r;
>    fi;
>  od;
> k:=k+1;
> fi;
> od;
> b;
>  end proc;
myLLL := proc(A)
local b, n, bb, i, mu, BB, k, r, j, mk, cc, mik, mkj, c, l;
    b := convert(LinearAlgebra:-Transpose(A), listlist);
    b := [seq(Vector(i), i = b)];
    n := nops(b);
    bb := b;
    for i from 2 to n do bb[i] := simplify(b[i] - add(((b[i]) . (bb[j]))*bb[j]/(bb[j]) . (bb[j]), j = 1 .. i - 1))
    end do;
    mu := Matrix(n, n, (i, j) -> (b[i]) . (bb[j])/(bb[j]) . (bb[j]));
    BB := [seq((bb[i]) . (bb[i]), i = 1 .. n)];
    k := 2;
    while k < n + 1 do
        if 0.5 < abs(mu[k, k - 1]) then
            r := round(mu[k, k - 1]);
            b[k] := b[k] - r*b[k - 1];
            for j to k - 2 do mu[k, j] := mu[k, j] - r*mu[k - 1, j] end do;
            mu[k, k - 1] := mu[k, k - 1] - r
        end if;
        if BB[k] < BB[k - 1]*(3/4 - mu[k, k - 1]^2) then
            mk := mu[k, k - 1];
            cc := BB[k] + mk^2*BB[k - 1];
            mu[k, k - 1] := mk*BB[k - 1]/cc;
            for i from k + 1 to n do
                mik := mu[i, k - 1];
                mu[i, k - 1] := mu[i, k - 1]*mu[k, k - 1] + mu[i, k]*BB[k]/cc;
                mu[i, k] := mik - mu[i, k]*mk
            end do;
            for j to k - 2 do mkj := mu[k, j]; mu[k, j] := mu[k - 1, j]; mu[k - 1, j] := mkj end do;
            c := b[k];
            b[k] := b[k - 1];
            b[k - 1] := c;
            BB[k] := BB[k - 1]*BB[k]/cc;
            BB[k - 1] := cc;
            k := max(2, k - 1)
        else
            for l from k - 2 by -1 to 1 do
                if 0.5 < abs(mu[k, l]) then
                    r := round(mu[k, l]);
                    b[k] := b[k] - r*b[l];
                    for j to l - 1 do mu[k, j] := mu[k, j] - r*mu[l, j] end do;
                    mu[k, l] := mu[k, l] - r
                end if
            end do;
            k := k + 1
        end if
    end do;
    b
end proc

> ##############
>  myLLL(M);
                                                    [ 0]  [ 1]  [1]
                                                    [  ]  [  ]  [ ]
                                                   [[ 0], [-1], [1]]
                                                    [  ]  [  ]  [ ]
                                                    [-1]  [ 0]  [0]

> #verification. lattice travaille sur les lignes
>  Transpose(Matrix(lattice(convert(Transpose(M),listlist))));
bytes used=4000144, alloc=3472772, time=0.16
                                                    [ 0     1    1]
                                                    [             ]
                                                    [ 0    -1    1]
                                                    [             ]
                                                    [-1     0    0]

> ##########
> #sac \'a dos
> s:=rand(200)():A:=[s]:for i from 1 to 6 do
> s:=s+rand(200)():A:=[op(A),s]: od;
                                                         s := 151

                                                      A := [81, 151]

                                                         s := 248

                                                   A := [81, 151, 248]

                                                         s := 411

                                                A := [81, 151, 248, 411]

                                                         s := 587

                                             A := [81, 151, 248, 411, 587]

                                                         s := 725
```

```
                                    A := [81, 151, 248, 411, 587, 725]

                                                  s := 810

                                A := [81, 151, 248, 411, 587, 725, 810]

>  M:=2001;w:=101;igcd(w,M);
                                                 M := 2001

                                                  w := 101

                                                     1

>  B:=[seq(modp(w*i,M),i=A)];
                            B := [177, 1244, 1036, 1491, 1258, 1189, 1770]

>  e:=[1,0,1,0,0,0,1];# le message a transmettre.
                                        e := [1, 0, 1, 0, 0, 0, 1]

>  c:=add(e[i]*B[i],i=1..nops(B));
                                                 c := 2983

>  M:=Matrix(nops(B)+1,nops(B)+1,(i,j)->if i=j-1 then 1 else 0 fi);
                                          [0    1    0    0    0    0    0    0]
                                          [                                     ]
                                          [0    0    1    0    0    0    0    0]
                                          [                                     ]
                                          [0    0    0    1    0    0    0    0]
                                          [                                     ]
                                          [0    0    0    0    1    0    0    0]
                                    M := [                                     ]
                                          [0    0    0    0    0    1    0    0]
                                          [                                     ]
                                          [0    0    0    0    0    0    1    0]
                                          [                                     ]
                                          [0    0    0    0    0    0    0    1]
                                          [                                     ]
                                          [0    0    0    0    0    0    0    0]

>  M[nops(B)+1,1]:=c;for i from 1 to nops(B) do M[nops(B)+1,i+1]:=B[i]
                                              M[8, 1] := 2983

>  od:M;
                          [   0      1       0       0       0       0       0       0]
                          [                                                         ]
                          [   0      0       1       0       0       0       0       0]
                          [                                                         ]
                          [   0      0       0       1       0       0       0       0]
                          [                                                         ]
                          [   0      0       0       0       1       0       0       0]
                          [                                                         ]
                          [   0      0       0       0       0       1       0       0]
                          [                                                         ]
                          [   0      0       0       0       0       0       1       0]
                          [                                                         ]
                          [   0      0       0       0       0       0       0       1]
                          [                                                         ]
                          [2983    177    1244    1036    1491    1258    1189    1770]

>  myLLL(M);# la premiere colonne est bien le message e
bytes used=8000328, alloc=4783252, time=0.31
bytes used=12000508, alloc=4979824, time=0.44
bytes used=16000676, alloc=4979824, time=0.56
                                  [1]  [ 0]  [-1]  [ 0]  [ 0]  [-1]  [ 1]  [ 3]
                                  [ ]  [  ]  [  ]  [  ]  [  ]  [  ]  [  ]  [  ]
                                  [0]  [ 0]  [-1]  [ 0]  [ 4]  [-2]  [ 1]  [-1]
                                  [ ]  [  ]  [  ]  [  ]  [  ]  [  ]  [  ]  [  ]
                                  [1]  [ 0]  [ 0]  [ 2]  [ 1]  [ 1]  [-2]  [-1]
                                  [ ]  [  ]  [  ]  [  ]  [  ]  [  ]  [  ]  [  ]
                                  [0]  [-2]  [ 1]  [-1]  [ 0]  [ 1]  [ 0]  [ 0]
                                 [[ ],  [ ], [ ],  [ ],  [ ],  [ ], [ ],  [ ]]
                                  [0]  [ 0]  [-1]  [ 0]  [-1]  [ 2]  [ 1]  [-1]
                                  [ ]  [  ]  [  ]  [  ]  [  ]  [  ]  [  ]  [  ]
                                  [0]  [ 0]  [ 1]  [ 1]  [ 0]  [-2]  [ 2]  [-1]
                                  [ ]  [  ]  [  ]  [  ]  [  ]  [  ]  [  ]  [  ]
                                  [1]  [ 0]  [ 0]  [-1]  [-1]  [ 0]  [ 0]  [-1]
                                  [ ]  [  ]  [  ]  [  ]  [  ]  [  ]  [  ]  [  ]
                                  [0]  [ 1]  [ 1]  [ 0]  [ 1]  [ 0]  [ 2]  [ 0]

>  #ou bien avec l'instruction maple:
>  lattice(convert(Transpose(M),listlist));# la premiere ligne est le message
[[1, 0, 1, 0, 0, 0, 1, 0], [0, 0, 0, -2, 0, 0, 0, 1], [-1, -1, 0, 1, -1, 1, 0, 1], [0, 0, 2, -1, 0, 1, -1, 0],

    [0, 4, 1, 0, -1, 0, -1, 1], [-1, -2, 1, 1, 2, -2, 0, 0], [1, 1, -2, 0, 1, 2, 0, 2], [3, -1, -1, 0, -1, -1, -1, 0]]

> quit
bytes used=18912752, alloc=4979824, time=0.67
```