```
    |\^/|     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);
>  with(LinearAlgebra);
[&x, Add, Adjoint, BackwardSubstitute, BandMatrix, Basis, BezoutMatrix, BidiagonalForm, BilinearForm,

    CharacteristicMatrix, CharacteristicPolynomial, Column, ColumnDimension, ColumnOperation, ColumnSpace,

    CompanionMatrix, ConditionNumber, ConstantMatrix, ConstantVector, Copy, CreatePermutation, CrossProduct,

    DeleteColumn, DeleteRow, Determinant, Diagonal, DiagonalMatrix, Dimension, Dimensions, DotProduct,

    EigenConditionNumbers, Eigenvalues, Eigenvectors, Equal, ForwardSubstitute, FrobeniusForm, GaussianElimination,

    GenerateEquations, GenerateMatrix, GetResultDataType, GetResultShape, GivensRotationMatrix, GramSchmidt,

    HankelMatrix, HermiteForm, HermitianTranspose, HessenbergForm, HilbertMatrix, HouseholderMatrix, IdentityMatrix,

    IntersectionBasis, IsDefinite, IsOrthogonal, IsSimilar, IsUnitary, JordanBlockMatrix, JordanForm, LA_Main,

    LUDecomposition, LeastSquares, LinearSolve, Map, Map2, MatrixAdd, MatrixExponential, MatrixFunction, MatrixInverse,

    MatrixMatrixMultiply, MatrixNorm, MatrixPower, MatrixScalarMultiply, MatrixVectorMultiply, MinimalPolynomial, Minor,

    Modular, Multiply, NoUserValue, Norm, Normalize, NullSpace, OuterProductMatrix, Permanent, Pivot, PopovForm,

    QRDecomposition, RandomMatrix, RandomVector, Rank, RationalCanonicalForm, ReducedRowEchelonForm, Row, RowDimension,

    RowOperation, RowSpace, ScalarMatrix, ScalarMultiply, ScalarVector, SchurForm, SingularValues, SmithForm, SubMatrix,

    SubVector, SumBasis, SylvesterMatrix, ToeplitzMatrix, Trace, Transpose, TridiagonalForm, UnitVector,

    VandermondeMatrix, VectorAdd, VectorAngle, VectorMatrixMultiply, VectorNorm, VectorScalarMultiply, ZeroMatrix,

    ZeroVector, Zip]

>  #with(Modular);
>  # pour pouvoir faire Mod d’une matrice. Pb ca n’existe pas avec maple7
>  # pour maple7 on recree Mod, k ne sert que pour etre compatible avec
>  # le Mod demaple9
>  Mod:=proc(n,A,k)
>  B:=Dimensions(A);
>  if nops([B])=1 then Vector(B,i->A[i] mod n) else
>  Matrix(B,(i,j)->A[i,j] mod n) fi
>  end proc;
Mod := proc(n, A, k)
local B;
    B := LinearAlgebra:-Dimensions(A);
    if nops([B]) = 1 then Vector(B, i -> A[i] mod n) else Matrix(B, (i, j) -> A[i, j] mod n) end if
end proc

>  #jeu1 du  jury
>  m:=Vector(7,0);
                                                             [0]
                                                             [ ]
                                                             [0]
                                                             [ ]
                                                             [0]
                                                             [ ]
                                                        m := [0]
                                                             [ ]
                                                             [0]
                                                             [ ]
                                                             [0]
                                                             [ ]
                                                             [0]

>  # ou bien m:=Matrix(7,1,0); et on rentre m[k,1]:=
>  H:=Matrix([[1,0,1,0,1,0,1],[0,1,1,0,0,1,1],[0,0,0,1,1,1,1]]);
                                              [1    0    1    0    1    0    1]
                                              [                              ]
                                         H := [0    1    1    0    0    1    1]
                                              [                              ]
                                              [0    0    0    1    1    1    1]

>  # entrer les valeurs non nulles (1=oui) selon les reponses Ex: m[1]:=1;
>  m[1]:=1;m[4]:=1;m[7]:=1;
                                                       m[1] := 1

                                                       m[4] := 1

                                                       m[7] := 1

>  w:=Mod(2,H.m,integer);#ok avec maple9
                                                             [0]
                                                             [ ]
                                                        w := [1]
                                                             [ ]
                                                             [0]

>  k:=4*w[3]+2*w[2]+w[1];
                                                         k := 2

>  if k<>0 then print("menti a la question:",k);m[k]:=m[k]+1 mod 2   else
>  print("pas menti"); fi:
                                               "menti a la question:", 2

>  r:=m[4]+m[3]*2+m[2]*4+m[1]*8:print("votre nombre est:",r);
```

```
                                                "votre nombre est:", 13

>  #le meme dans une procedure:
>  #m est le vecteur reponse: 1 pour oui a la question
>  jeu:=proc(m)
>  H:=Matrix([[1,0,1,0,1,0,1],[0,1,1,0,0,1,1],[0,0,0,1,1,1,1]]);
>  w:=Mod(2,H.m,integer);
>  k:=4*w[3]+2*w[2]+w[1];
>  if k<>0 then print("menti a la question:",k);m[k]:=m[k]+1 mod 2   else
>  print("pas menti"); fi:
>  r:=m[4]+m[3]*2+m[2]*4+m[1]*8:print("votre nombre est:",r);
>  end proc;
jeu := proc(m)
local H, w, k, r;
    H := Matrix([[1, 0, 1, 0, 1, 0, 1], [0, 1, 1, 0, 0, 1, 1], [0, 0, 0, 1, 1, 1, 1]]);
    w := Mod(2, H . m, integer);
    k := 4*w[3] + 2*w[2] + w[1];
    if k <> 0 then print("menti a la question:", k); m[k] := (m[k] + 1) mod 2 else print("pas menti") end if;
    r := m[4] + 2*m[3] + 4*m[2] + 8*m[1];
    print("votre nombre est:", r)
end proc

>  f:=(i,j)->floor(j/2^(i-1)) mod 2;
                                                                  j
                                          f := (i, j) -> floor(--------) mod 2
                                                                (i - 1)
                                                               2

>  H:=Matrix(3,7,f);
                                              [1    0    1    0    1    0    1]
                                              [                              ]
                                         H := [0    1    1    0    0    1    1]
                                              [                              ]
                                              [0    0    0    1    1    1    1]

>  #On trouve une famille generatrice du code:
>  noyau:=Nullspace(H) mod 2;
                                                       [1]  [1]  [0]  [1]
                                                       [ ]  [ ]  [ ]  [ ]
                                                       [1]  [0]  [1]  [1]
                                                       [ ]  [ ]  [ ]  [ ]
                                                       [1]  [0]  [0]  [0]
                                                       [ ]  [ ]  [ ]  [ ]
                                             noyau := {[0], [1], [1], [1]}
                                                       [ ]  [ ]  [ ]  [ ]
                                                       [0]  [1]  [0]  [0]
                                                       [ ]  [ ]  [ ]  [ ]
                                                       [0]  [0]  [1]  [0]
                                                       [ ]  [ ]  [ ]  [ ]
                                                       [0]  [0]  [0]  [1]

>  #On converti le resultat en une matrice.
>  C:=Matrix(convert(noyau,list));
                                                     [1    1    0    1]
                                                     [                ]
                                                     [1    0    1    1]
                                                     [                ]
                                                     [1    0    0    0]
                                                     [                ]
                                                C := [0    1    1    1]
                                                     [                ]
                                                     [0    1    0    0]
                                                     [                ]
                                                     [0    0    1    0]
                                                     [                ]
                                                     [0    0    0    1]

>  #On verifie:
>  Mod(2,H.C,integer);
                                                   [0    0    0    0]
                                                   [                ]
                                                   [0    0    0    0]
                                                   [                ]
                                                   [0    0    0    0]

>  subs({y=1,x=3},[x,y]);
                                                         [3, 1]

#Nullspace,Linsolve() mod 2,subs
>  f:=(i,j)->if i=j then 1  else 0 fi;
                     f := proc(i, j) option operator, arrow; if i = j then 1 else 0 end if end proc

>  #On insere H dans une matrice plus  grande.
>  syst:=proc(i)
>  HE:=Matrix(4,3,f).H;
>  HE[4,i]:=1;
>  HE;
>  end proc;
                   syst := proc(i) local HE; HE := (Matrix(4, 3, f)) . H; HE[4, i] := 1; HE end proc

>  sol:=proc(k)
>  j:=1;
>  X:=Linsolve(syst(k),Vector([0,0,0,1])) mod 2;
>  for i from 1 to 7 do if (X<>subs(_t[i]=0,X)) then X:=subs(_t[i]=_t[j],X);
>  j:=j+1; fi; od;
>  X;
>  end proc;
sol := proc(k)
local j, X, i;
    j := 1;
    X := Linsolve(syst(k), Vector([0, 0, 0, 1])) mod 2;
    for i to 7 do if X <> subs(_t[i] = 0, X) then X := subs(_t[i] = _t[j], X); j := j + 1 end if end do;
    X
end proc

>  #oui a la question k veut dire etre dans le code et m[i]=1, donc etre solution de syst(k)
>  S:=Linsolve(syst(3),Vector([0,0,0,1])) mod 2;
                                                  [  1 + _t[5] + _t[7]  ]
                                                  [                     ]
                                                  [  1 + _t[6] + _t[7]  ]
                                                  [                     ]
                                                  [          1          ]
                                                  [                     ]
                                             S := [_t[5] + _t[6] + _t[7]]
                                                  [                     ]
                                                  [        _t[5]        ]
                                                  [                     ]
                                                  [        _t[6]        ]
                                                  [                     ]
                                                  [        _t[7]        ]

>  Mod(2, subs({_t[5]=1,_t[6]=0,_t[7]=0}, S),integer);
                                                         [0]
                                                         [ ]
                                                         [1]
                                                         [ ]
                                                         [1]
                                                         [ ]
                                                         [1]
                                                         [ ]
                                                         [1]
                                                         [ ]
                                                         [0]
                                                         [ ]
                                                         [0]

>  question:=proc(k)
>  S:=sol(k);
>  f:=(a,b,c)->Mod(2, subs({_t[1]=a,_t[2]=b,_t[3]=c}, S),integer);
# [1] pour retirer les []
>  l:=seq(seq(seq((Matrix([[8,4,2,1,0,0,0]]).f(a,b,c))[1],a=0..1),b=0..1),c=0..1);
>  sort([l]);
>  end proc;
question := proc(k)
local S, f, l;
    S := sol(k);
    f := (a, b, c) -> Mod(2, subs({_t[1] = a, _t[2] = b, _t[3] = c}, S), integer);
    l := seq(seq(seq(((Matrix([[8, 4, 2, 1, 0, 0, 0]])) . (f(a, b, c)))[1], a = 0 .. 1), b = 0 .. 1), c = 0 .. 1);
    sort([l])
end proc

>  for k from 1 to 7 do print("est il dans",question(k));od;
                                     "est il dans", [8, 9, 10, 11, 12, 13, 14, 15]

                                      "est il dans", [4, 5, 6, 7, 12, 13, 14, 15]

                                      "est il dans", [2, 3, 6, 7, 10, 11, 14, 15]

                                       "est il dans", [1, 3, 5, 7, 9, 11, 13, 15]

bytes used=4000068, alloc=3472772, time=0.17
                                       "est il dans", [1, 2, 4, 7, 9, 10, 12, 15]

                                       "est il dans", [1, 2, 5, 6, 8, 11, 12, 15]

                                       "est il dans", [1, 3, 4, 6, 8, 10, 13, 15]

>  prodlist:=(E,F)->[seq(seq([i,j],j=F),i=E)];
                                 prodlist := (E, F) -> [seq(seq([i, j], j = F), i = E)]

>  powerlist:=proc(E,n)
>   if n=0 then [[]] else
>   aa:=prodlist(E,powerlist(E,n-1)) ; map(a->[a[1],op(a[2])],aa) fi
>   end proc;
powerlist := proc(E, n)
local aa;
    if n = 0 then [[]] else aa := prodlist(E, powerlist(E, n - 1)); map(a -> [a[1], op(a[2])], aa) end if
end proc

>   # pour enlever les [] faire op
>
>  questionbis:=proc(k)
>  S:=sol(k);
>  L:=powerlist([0,1],3);
>  f:=(a,b,c)->Mod(2, subs({_t[1]=a,_t[2]=b,_t[3]=c},S),integer);
>  # [1] pour retirer les []
>  l:=seq((Matrix([[8,4,2,1,0,0,0]]).f(op(A)))[1],A=L);
>  sort([l]);
>  end proc;
questionbis := proc(k)
local S, L, f, l;
    S := sol(k);
    L := powerlist([0, 1], 3);
    f := (a, b, c) -> Mod(2, subs({_t[1] = a, _t[2] = b, _t[3] = c}, S), integer);
    l := seq(((Matrix([[8, 4, 2, 1, 0, 0, 0]])) . (f(op(A))))[1], A = L);
    sort([l])
end proc

>  for k from 1 to 7 do print("est il dans",questionbis(k));od;
                                     "est il dans", [8, 9, 10, 11, 12, 13, 14, 15]

                                      "est il dans", [4, 5, 6, 7, 12, 13, 14, 15]
```

```
                                          "est il dans", [2, 3, 6, 7, 10, 11, 14, 15]

                                           "est il dans", [1, 3, 5, 7, 9, 11, 13, 15]

                                           "est il dans", [1, 2, 4, 7, 9, 10, 12, 15]

                                           "est il dans", [1, 2, 5, 6, 8, 11, 12, 15]

                                           "est il dans", [1, 3, 4, 6, 8, 10, 13, 15]

>  l:={0,1,j,1+j};
                                                    l := {0, 1, j, 1 + j}

>  # Attention, si on a charge Modular, Transpose est modifiee, faire alors:
>  #Transpose(2,Matrix([seq(seq([u,v],v=l),u=l)]));
>  Transpose(Matrix([seq(seq([u,v],v=l),u=l)]));
bytes used=8000256, alloc=4717728, time=0.29
       [0    0    0      0      1    1    1      1      j    j    j      j      1 + j    1 + j    1 + j    1 + j]
       [                                                                                                       ]
       [0    1    j    1 + j    0    1    j    1 + j    0    1    j    1 + j      0        1        j      1 + j]

>  Hb:='Hb';
                                                          Hb := Hb

>  Hb:=Matrix([[0,seq(1,i=l)],[1,seq(i,i=l)]]);
                                                [0    1    1    1      1  ]
                                          Hb := [                         ]
                                                [1    0    1    j    1 + j]

>  f:=(i,j)->if (i=j+1) then 1 else 0 fi;
                 f := proc(i, j) option operator, arrow; if i = j + 1 then 1 else 0 end if end proc

>  T:=Matrix(3,2,f);
                                                         [0    0]
                                                         [      ]
                                                    T := [1    0]
                                                         [      ]
                                                         [0    1]

>  A:=convert(Transpose(T.Hb),listlist);
                            A := [[0, 0, 1], [0, 1, 0], [0, 1, 1], [0, 1, j], [0, 1, 1 + j]]

>  op(A);
                               [0, 0, 1], [0, 1, 0], [0, 1, 1], [0, 1, j], [0, 1, 1 + j]

>  # la generalisation du code sur F4 est le noyau de:
>  H4:=Transpose(Matrix([op(A),seq(seq([1,u,v],v=l),u=l)]));
           [0 ,  0 ,  0 ,  0 ,  0 ,  1 ,  1 ,  1 ,  1 ,  1 ,  1 ,  1 ,  1 ,  1 ,  1 ,  1 ,  1 ,  1 ,  1 ,  1 ,  1
           [
     H4 := [0 ,  1 ,  1 ,  1 ,  1 , 0 , 0 , 0 , 0 , 1 , 1 , 1 , 1 , j , j , j , j , 1 + j , 1 + j , 1 + j , 1 + j
           [
           [1 , 0 , 1 , j , 1 + j , 0 , 1 , j , 1 + j , 0 , 1 , j , 1 + j , 0 , 1 , j , 1 + j , 0 , 1 , j , 1 + j

>  #Pb Comment trouver un noyau dans F4
>  Nullspace(H4) mod 2;#on a de la chance, ca donne bien 18 vecteurs du noyau, mais on n'est pas sur que la famil
libre sur F4, il faudrait verifier que ses facteurs invariants ne sont pas divisibles par j^2+j+1
 [1]  [  1  ]  [  0  ]  [j]  [1]  [j]  [1 + j]  [1]  [1 + j]  [j]  [  j  ]  [0]  [1]  [1 + j]  [1 + j]  [0]  [1 +
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [1]  [1 + j]  [1 + j]  [1]  [1]  [1]  [  j  ]  [0]  [1 + j]  [j]  [1 + j]  [1]  [j]  [  0  ]  [  1  ]  [j]  [  1
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [1]  [  0  ]  [  0  ]  [0]  [0]  [0]  [  0  ]  [0]  [  0  ]  [0]  [  0  ]  [0]  [0]  [  0  ]  [  0  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [0]  [0]  [1]  [  0  ]  [0]  [  0  ]  [0]  [  0  ]  [0]  [0]  [  0  ]  [  0  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [0]  [0]  [0]  [  0  ]  [0]  [  0  ]  [0]  [  0  ]  [0]  [0]  [  0  ]  [  1  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  1  ]  [  1  ]  [1]  [1]  [0]  [  1  ]  [1]  [  1  ]  [1]  [  1  ]  [1]  [1]  [  1  ]  [  0  ]  [1]  [  1
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [0]  [0]  [0]  [  0  ]  [1]  [  0  ]  [0]  [  0  ]  [0]  [0]  [  0  ]  [  0  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [0]  [0]  [0]  [  0  ]  [0]  [  0  ]  [0]  [  0  ]  [0]  [0]  [  0  ]  [  0  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [0]  [0]  [0]  [  0  ]  [0]  [  0  ]  [0]  [  0  ]  [0]  [0]  [  1  ]  [  0  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [0]  [0]  [0]  [  0  ]  [0]  [  0  ]  [0]  [  0  ]  [1]  [0]  [  0  ]  [  0  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
{[0], [  0  ], [  0  ], [0], [1], [0], [  0  ], [0], [  0  ], [0], [  0  ], [0], [0], [  0  ], [  0  ], [0], [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [1]  [0]  [0]  [  0  ]  [0]  [  0  ]  [0]  [  0  ]  [0]  [0]  [  0  ]  [  0  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [0]  [0]  [0]  [  0  ]  [0]  [  0  ]  [0]  [  0  ]  [0]  [0]  [  0  ]  [  0  ]  [0]  [  1
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [0]  [0]  [0]  [  0  ]  [0]  [  0  ]  [0]  [  0  ]  [0]  [0]  [  0  ]  [  0  ]  [1]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [0]  [0]  [0]  [  0  ]  [0]  [  0  ]  [0]  [  0  ]  [0]  [1]  [  0  ]  [  0  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [0]  [0]  [0]  [  0  ]  [0]  [  0  ]  [1]  [  0  ]  [0]  [0]  [  0  ]  [  0  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [0]  [0]  [0]  [  1  ]  [0]  [  0  ]  [0]  [  0  ]  [0]  [0]  [  0  ]  [  0  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  1  ]  [0]  [0]  [0]  [  0  ]  [0]  [  0  ]  [0]  [  0  ]  [0]  [0]  [  0  ]  [  0  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  1  ]  [  0  ]  [0]  [0]  [0]  [  0  ]  [0]  [  0  ]  [0]  [  0  ]  [0]  [0]  [  0  ]  [  0  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [0]  [0]  [0]  [  0  ]  [0]  [  0  ]  [0]  [  1  ]  [0]  [0]  [  0  ]  [  0  ]  [0]  [  0
 [ ]  [     ]  [     ]  [ ]  [ ]  [ ]  [     ]  [ ]  [     ]  [ ]  [     ]  [ ]  [ ]  [     ]  [     ]  [ ]  [
 [0]  [  0  ]  [  0  ]  [0]  [0]  [0]  [  0  ]  [0]  [  1  ]  [0]  [  0  ]  [0]  [0]  [  0  ]  [  0  ]  [0]  [  0

    [j]
    [ ]
    [0]
    [ ]
    [0]
    [ ]
    [0]
    [ ]
    [0]
    [ ]
    [1]
    [ ]
    [0]
    [ ]
    [1]
    [ ]
    [0]
    [ ]
    [0]
    [ ]
    [0]}
    [ ]
    [0]
    [ ]
    [0]
    [ ]
    [0]
    [ ]
    [0]
    [ ]
    [0]
    [ ]
    [0]
    [ ]
    [0]
    [ ]
    [0]
    [ ]
    [0]
    [ ]
    [0]

> quit
bytes used=8529168, alloc=4717728, time=0.31
```