```
> restart;
```

CONSTRUCTION DE CARRES MAGIQUES

Testons si un carre de taille n est pandiagonal d'ordre n

```
> with(LinearAlgebra):
> test:=proc(a)
  local M,k,i,j,e,S,n,l,A;
  M:=a; n:=Dimension(a)[1]; S:=n*(n^2-1)/2;
  l:=seq(i,i=0..n^2-1);l:={l};
  e:={seq(seq(M[i,j],i=1..n),j=1..n)};
  if e<>l then return print ("Ce n'est pas un carre eulerien
  pandiagonal"); fi;
  for i from 1 to n do
       if add(M[i,j],j=1..n)<>S then return print ("Ce n'est
  pas un carre eulerien pandiagonal") fi;
       if add(M[j,i],j=1..n)<>S then return print ("Ce n'est
  pas un carre eulerien pandiagonal"); fi;
       if add(M[j,(j+i) mod n +1],j=1..n)<>S then return print
  ("Ce n'est pas un carre eulerien pandiagonal"); fi;
       if add(M[j,(-1-i-j) mod n +1],j=1..n)<>S then return
  print ("Ce n'est pas un carre eulerien pandiagonal"); fi;
  end do;
  print(M);print("est un carre eulerien pandiagonal");
  return true;
  end proc:
>
> a:=Matrix([[0,24,18,12,6],[13,7,1,20,19],[21,15,14,8,2],[9,3,
  22,16,10],[17,11,5,4,23]]);
```

$$a := \begin{bmatrix} 0 & 24 & 18 & 12 & 6 \\ 13 & 7 & 1 & 20 & 19 \\ 21 & 15 & 14 & 8 & 2 \\ 9 & 3 & 22 & 16 & 10 \\ 17 & 11 & 5 & 4 & 23 \end{bmatrix}$$

```
> test(a);
```

$$\begin{bmatrix} 0 & 24 & 18 & 12 & 6 \\ 13 & 7 & 1 & 20 & 19 \\ 21 & 15 & 14 & 8 & 2 \\ 9 & 3 & 22 & 16 & 10 \\ 17 & 11 & 5 & 4 & 23 \end{bmatrix}$$

"est un carre eulerien pandiagonal"

true

```
> b:=Matrix([[0,2,18,12,6],[13,7,1,20,19],[21,15,14,8,2],[9,3,2
  2,16,10],[17,11,5,4,23]]);
```

$$b := \begin{bmatrix} 0 & 2 & 18 & 12 & 6 \\ 13 & 7 & 1 & 20 & 19 \\ 21 & 15 & 14 & 8 & 2 \\ 9 & 3 & 22 & 16 & 10 \\ 17 & 11 & 5 & 4 & 23 \end{bmatrix}$$

```
> test(b);
```

"Ce n'est pas un carre eulerien pandiagonal"

```
>
```

Algorithme du cavalier

Un premier programme en codant "case par case" selon l'algorithme du cavalier

```
> with(LinearAlgebra):
  caval1:=proc(n)
  local M,i,j,k,l,N;
  M:=Array(0..n-1,0..n-1,fill=-1);
  N:=Matrix(n);
  M[0,0]:=0;
  i:=0;j:=0;
  for k from 1 to n^2-1 do
       if M[irem(i+1,n),irem(j+2,n)]=-1 then i:=i+1 mod
  n;j:=j+2 mod n;M[i,j]:=k;N[i+1,j+1]:=k
       else j:=j-1 mod n; M[i,j]:=k;fi;N[i+1,j+1]:=k;

  end do;
  test(N);
  end proc:
```

```
> caval1(5);
```

$$\begin{bmatrix} 0 & 24 & 18 & 12 & 6 \\ 13 & 7 & 1 & 20 & 19 \\ 21 & 15 & 14 & 8 & 2 \\ 9 & 3 & 22 & 16 & 10 \\ 17 & 11 & 5 & 4 & 23 \end{bmatrix}$$

"est un carre eulerien pandiagonal"

true

```
> caval1(7);
```

$$\begin{bmatrix} 0 & 48 & 40 & 32 & 24 & 16 & 8 \\ 17 & 9 & 1 & 42 & 41 & 33 & 25 \\ 34 & 26 & 18 & 10 & 2 & 43 & 35 \\ 44 & 36 & 28 & 27 & 19 & 11 & 3 \\ 12 & 4 & 45 & 37 & 29 & 21 & 20 \\ 22 & 14 & 13 & 5 & 46 & 38 & 30 \\ 39 & 31 & 23 & 15 & 7 & 6 & 47 \end{bmatrix}$$

"est un carre eulerien pandiagonal"

true

Un deuxieme programme a partir du calcul des coefficients d'un carre construit a l'aide de l'algorithme du cavalier

```
> caval2:=proc(n)
  local M,N,r,q,i,j,k;
  M:=Array(0..n-1,0..n-1,fill=0);
  N:=Matrix(n);
  for k from 1 to n^2-1 do
        r:=k mod n;
        q:=iquo(k,n);
        i:=r-q mod n;
        j:=2*r-3*q mod n;
        M[i,j]:=k; N[i+1,j+1]:=k;
  od;
  test(N);
  end proc:
> caval2(5);
```

$$\begin{bmatrix} 0 & 24 & 18 & 12 & 6 \\ 13 & 7 & 1 & 20 & 19 \\ 21 & 15 & 14 & 8 & 2 \\ 9 & 3 & 22 & 16 & 10 \\ 17 & 11 & 5 & 4 & 23 \end{bmatrix}$$

"est un carre eulerien pandiagonal"

true

```
> caval2(7);
```

$$\begin{bmatrix} 0 & 48 & 40 & 32 & 24 & 16 & 8 \\ 17 & 9 & 1 & 42 & 41 & 33 & 25 \\ 34 & 26 & 18 & 10 & 2 & 43 & 35 \\ 44 & 36 & 28 & 27 & 19 & 11 & 3 \\ 12 & 4 & 45 & 37 & 29 & 21 & 20 \\ 22 & 14 & 13 & 5 & 46 & 38 & 30 \\ 39 & 31 & 23 & 15 & 7 & 6 & 47 \end{bmatrix}$$

"est un carre eulerien pandiagonal"

true

Generalisation de la methode du cavalier

```
> with(combinat,choose):
> const:=proc(L,k,n)
  local M,N,r,q,i,j;
  M:=Array(0..n-1,0..n-1,fill=0);
  N:=Matrix(n);
  for i from 0 to n-1 do
       for j from 0 to n-1 do
            M[i,j]:=L[(k*i-j) mod n +1]; N[i+1,j+1]:=M[i,j];
       od;
  od;
  N;
  end proc:
> const([0,1,2,3,4],2,5);
>
```

$$\begin{bmatrix} 0 & 4 & 3 & 2 & 1 \\ 2 & 1 & 0 & 4 & 3 \\ 4 & 3 & 2 & 1 & 0 \\ 1 & 0 & 4 & 3 & 2 \\ 3 & 2 & 1 & 0 & 4 \end{bmatrix}$$

```
> with(LinearAlgebra):
> prod:=proc(M,N)
  local n,i,j,P;
  n:=Dimension(M)[1];
  P:=Matrix(n);
  for i from 1 to n do
       for j from 1 to n do
```

```
            P[i,j]:=n*M[i,j]+N[i,j];
        od;
    od;
    return P;
    end proc;
```

prod := proc(M, N)
local n, i, j, P;
n := LinearAlgebra:-Dimension(M)[1];
P := Matrix(n);
for i to n do for j to n do P[i, j] := n*M[i, j] + N[i, j] end do end do;
return P
end proc

```
> genint:=proc(n)
  local L,k,i;
  L:=NULL;
  for i from 2 to n-2 do
       if igcd(i-1,n)=1 and igcd(i,n)=1 and igcd(i+1,n)=1 then
           k:=k+1;
           L:=L,i;
       fi;
  od;
  L:=[L];
  return L;
  end proc;
```

genint := proc(n)
local L, k, i;
L := NULL;
for i from 2 to n − 2 do
if igcd(i − 1, n) = 1 and igcd(i + 1, n) = 1 then k := k + 1; L := L, i end if
end do;
L := [L];
return L
end proc

```
> carre:=proc(n)
  local i,k,L,p1,p2,r,j,M,tc,c;
  p1:=0;p2:=0;r:=0;
  L:=genint(n);
  c:=choose(L,2);
  tc:=nops(c);
  if nops(L)<=1 then return printf("cette construction ne
  permet pas de produire un carre de taille %n");
  else
       for i from 1 to tc do
            if igcd((c[i][2]-c[i][1]),n)=1 then
                 p1:=c[i][1];
                 p2:=c[i][2];fi;
       od;
```

```
fi;
if p1=0 then return printf("cette construction ne permet pas
de produire un carre de taille %n");
else
     return
prod(const([seq(k,k=0..n-1)],p1,n),const([seq(k,k=0..n-1)],p2
,n));
fi;
end proc:
> carre(5);
```

$$\begin{bmatrix} 0 & 24 & 18 & 12 & 6 \\ 13 & 7 & 1 & 20 & 19 \\ 21 & 15 & 14 & 8 & 2 \\ 9 & 3 & 22 & 16 & 10 \\ 17 & 11 & 5 & 4 & 23 \end{bmatrix}$$

```
> test(carre(7));
```

$$\begin{bmatrix} 0 & 48 & 40 & 32 & 24 & 16 & 8 \\ 33 & 25 & 17 & 9 & 1 & 42 & 41 \\ 10 & 2 & 43 & 35 & 34 & 26 & 18 \\ 36 & 28 & 27 & 19 & 11 & 3 & 44 \\ 20 & 12 & 4 & 45 & 37 & 29 & 21 \\ 46 & 38 & 30 & 22 & 14 & 13 & 5 \\ 23 & 15 & 7 & 6 & 47 & 39 & 31 \end{bmatrix}$$

"est un carre eulerien pandiagonal"

true

```
>
```