

Das Problem der 36 Offiziere

Annika Bölte & Sarah Lüke

22.01.2003

```
> restart;
```

```
with(LinearAlgebra):
```

Definition: Eine n mal n Matrix mit Einträgen in $\{1, \dots, n\}$ nennen wir n -Quadrat. Ein n -Quadrat heißt **lateinisches** Quadrat, genau dann wenn jeder Eintrag in jeder Zeile und in jeder Spalte genau einmal vorkommt. Zwei Quadrate heißen **orthogonal**, wenn beim Übereinanderlegen jedes (geordnete) Paar genau einmal vorkommt.

Wir können immer ohne Einschränkung annehmen, dass die erste Zeile eines lateinischen n -Quadrates $1, 2, 3, \dots, n$ lautet. Denn in einem lateinischen Quadrat können die Zeilen und die Spalten vertauscht werden, ohne dass die Eigenschaft, dass in jeder Zeile und jeder Spalte jeder Eintrag nur genau einmal vorkommt, verlorenght. Solch ein n -Quadrat, dessen erste Zeile $1, 2, 3, \dots, n$ lautet nennen wir **normiert**.

Gesucht: Alle normierten 3-Quadrate

```
> Q1:=<<1 | 2 | 3> , <2 | 3 | 1> , <3 | 1 | 2>>;
```

$$Q1 := \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

```
> Q2:=<<1 | 2 | 3> , <3 | 1 | 2> , <2 | 3 | 1>>;
```

$$Q2 := \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix}$$

Q2 ergibt sich aus Q1, indem man die letzten beiden Zeilen vertauscht.

Es gibt keine weiteren normierten 3-Quadrate!

Wenn man die Quadrate direkt übereinanderlegt, entstehen folgende Paare (1,1), (2,2), (3,3), (2,3), (3,1), (1,2), (3,2), (1,3), (2,1) => sie sind orthogonal zueinander.

Hier eine Prozedur, die testet, ob ein Quadrat lateinisch ist:

```
> islatin := proc(A)
  local n,z,s,i,k;
  n := RowDimension(A);
  for z from 1 to n do
    for s from 1 to n do
      for i from 1 to n while i<> z do
        if A [z,s] = A[i,s] then return false end if;
      end do;
      for k from 1 to n while k <> s do
        if A[z,s]=A[z,k] then return false end if;
      end do;
    end do;
  end do;
```

```

    end do;
end do;
return true;
end proc:

```

```
> islatin(Q1);
```

```
Error, (in LinearAlgebra:-RowDimension) expects its 1st argument, A, to be of
type Matrix, but received Q1
```

Die Methode Listentest überprüft, ob das Element "Elem" in der Liste "L" enthalten ist. Listentest wird für die nachfolgende Prozedur "orthogonal" benötigt.

```
>
Listentest := proc (L,Elem)
local i;
  for i from 1 to nops(L) do
    if L[i]= Elem then return true; end if;
  end do;
  return false;
end proc:

```

Diese Prozedur testet, ob zwei Quadrate A und B orthogonal sind.

```
>
orthogonal := proc (A,B)
local dA,dB,L,z,s,El;
  dA := RowDimension(A);
  dB := RowDimension(B);
  if dA <> ColumnDimension(A) or dB <> ColumnDimension(B) then
return false; end if;
  L := [];
  if dA <> dB then return false; end if;
  for z from 1 to dA do
    for s from 1 to dA do
      El:=[A[z,s],B[z,s]];
      if Listentest (L,El)=false then L:=[op(L), El];
      else return false; end if;
    end do;
  end do;
  return true;
end proc:

```

```
> orthogonal(Q1, Q2);
```

```
      true
```

```
> with(plots):
  with(plottools):
```

```
Warning, the name arrow has been redefined
```

```
Warning, the name arrow has been redefined
```

Die Methode plotmatrix zeichnet zwei n mal n Matrizen, indem sie ein Quadrat zeichnet, das in n mal n kleine Quadrate unterteilt ist, die jeweils wieder kleine Quadrate enthalten. Diese Quadrate werden je nachdem, welchen Eintrag die Matrix an dieser Stelle hat, unterschiedlich gefärbt.

```
plotmatrix:=proc(M1,M2)
  local i,j,L,s,p,d1,d2;
  L:=NULL;
```

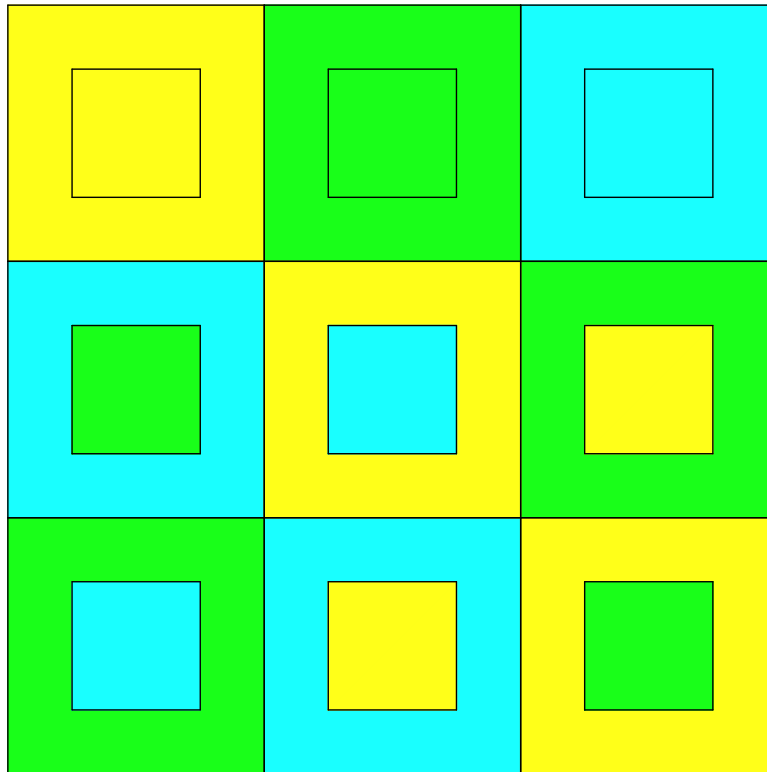
```

d1:=RowDimension(M1);
d2:=RowDimension(M2);
  for p from 1 to d1 do
  for i from 1 to d1 do
    for j from 1 to ColumnDimension(M1) do
      if M1[i,j]=p then s:=p/((d1-1)*3);
L:=L,rectangle([j-1/4,-i-1/4],[j+1/4,-i+1/4],color=COLOR(HUE,s));e
nd if;
      end do;
    end do;
  end do;
for p from 1 to d2 do
  for i from 1 to d2 do
    for j from 1 to ColumnDimension(M2) do
      if M2[i,j]=p then s:=p/((d2-1)*3);
L:=L,rectangle([j-1/2,-i-1/2],[j+1/2,-i+1/2],color=COLOR(HUE,s));e
nd if;
      end do;
    end do;
  end do;
plots[display]([L],axes=NONE,scaling=constrained);
end proc:

```

Wenn jede Farbkonstellation genau einmal vorkommt, bedeutet das, dass die beiden Matrizen orthogonal sind.

```
> plotmatrix(Q1,Q2);
```

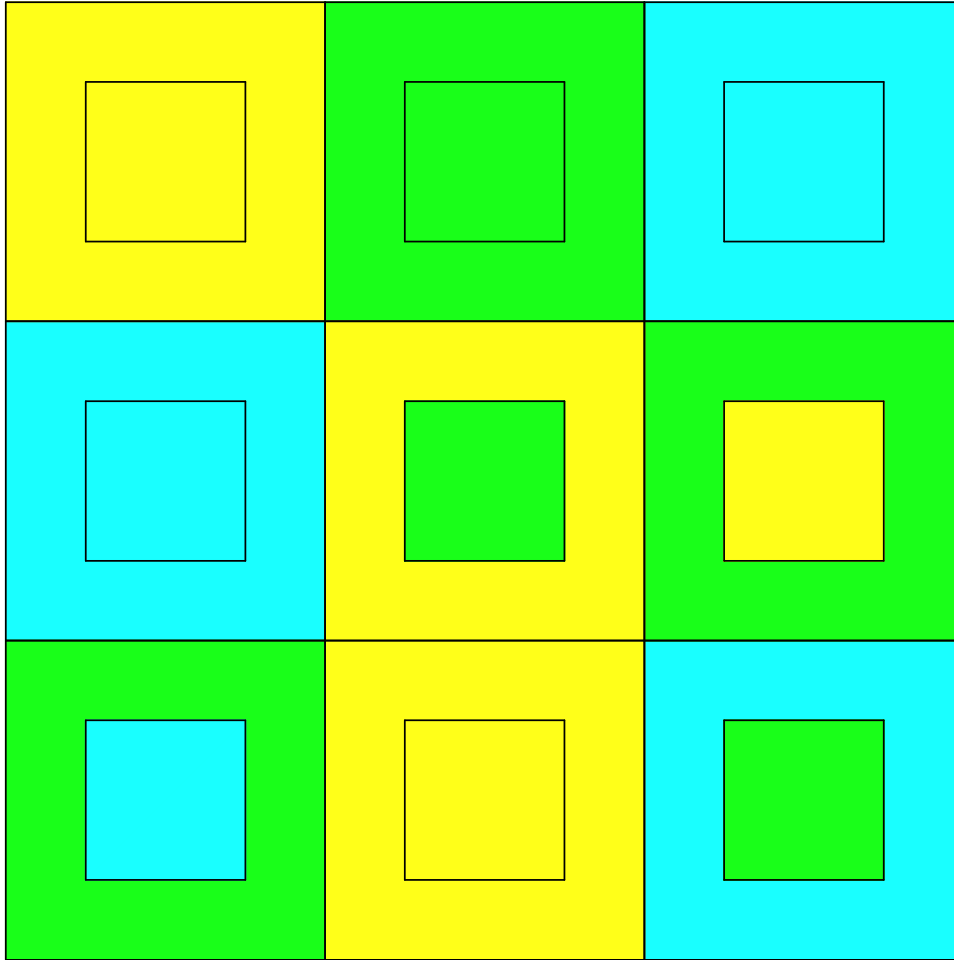


Beispiel für zwei Matrizen, die nicht orthogonal sind:

```
> A:=<<1 | 2 | 3> , <3 | 2 | 1> , <3 | 1 | 2>>;
  B:=<<1 | 2 | 3> , <3 | 1 | 2> , <2 | 1 | 3>>;
  plotmatrix(A,B);
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \\ 3 & 1 & 2 \end{bmatrix}$$

$$B := \begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 1 & 3 \end{bmatrix}$$



Behauptung: Ein Quadrat ist genau dann lateinisch, wenn es zu den beiden Quadraten Z und S orthogonal ist.

Die Matrix Z hat in der i -ten Zeile nur i . Die Matrix S hat in der j -ten Spalte nur j . Angenommen, die Matrix M ist orthogonal zu Z und S . Die geordneten Paare, die beim Übereinanderlegen von M und Z entstehen, beginnen in der ersten Zeile immer mit einer 1. Da M zu Z orthogonal ist, kommt jedes geordnete Paar nur genau einmal vor. Das bedeutet, dass die zweiten Komponenten in der ersten Zeile (der übereinandergelegten Matrizen M und Z), also gerade die Elemente der ersten Zeile von M paarweise verschieden sind, d.h. in dieser Zeile kommt jedes Element genau einmal vor. Genauso lässt sich aus der Orthogonalität von M zu S folgern, dass bei M in jeder Spalte jedes Element nur genau einmal vorkommt. Das bedeutet, dass M lateinisch ist.

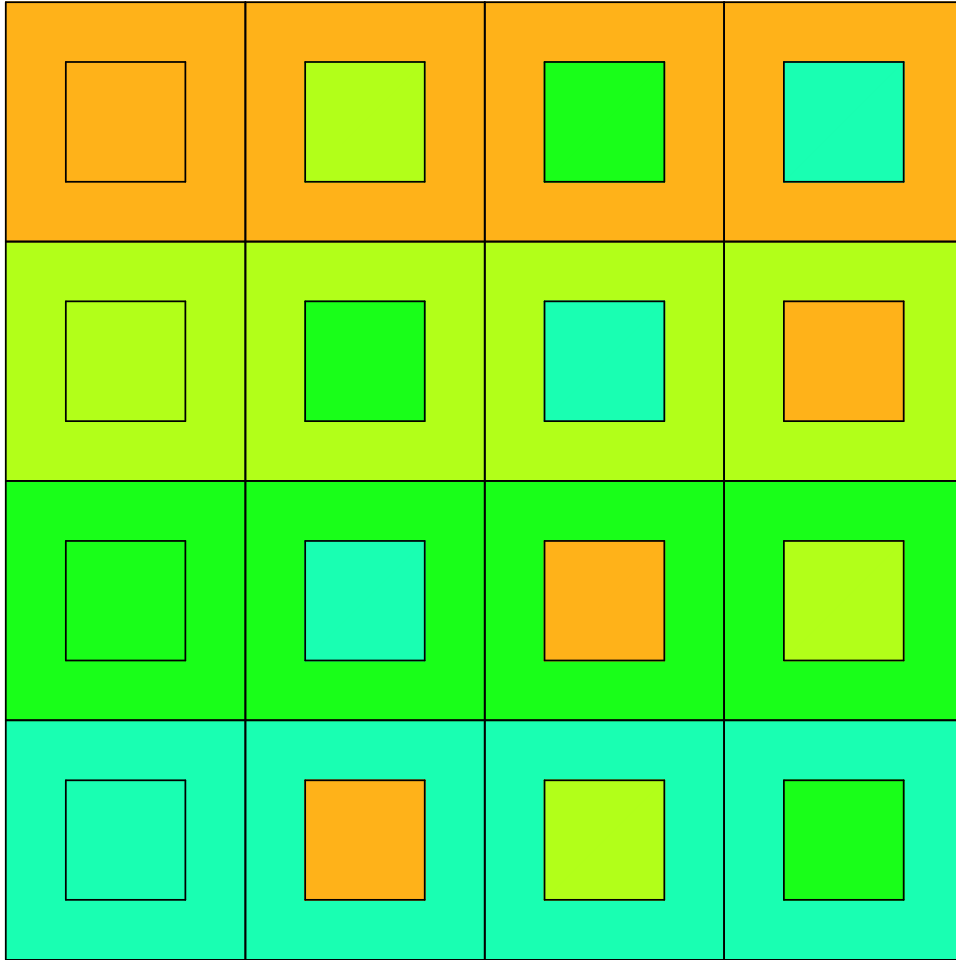
```
> M:=<<1 | 2 | 3 | 4> , <2 | 3 | 4 | 1> , <3 | 4 | 1 | 2> , <4 | 1 | 2 | 3>>;
```

$$M := \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{bmatrix}$$

```
> Z := <<1 | 1 | 1 | 1> , <2 | 2 | 2 | 2> , <3 | 3 | 3 | 3> , <4 | 4 | 4  
| 4>> i;
```

$$Z := \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix}$$

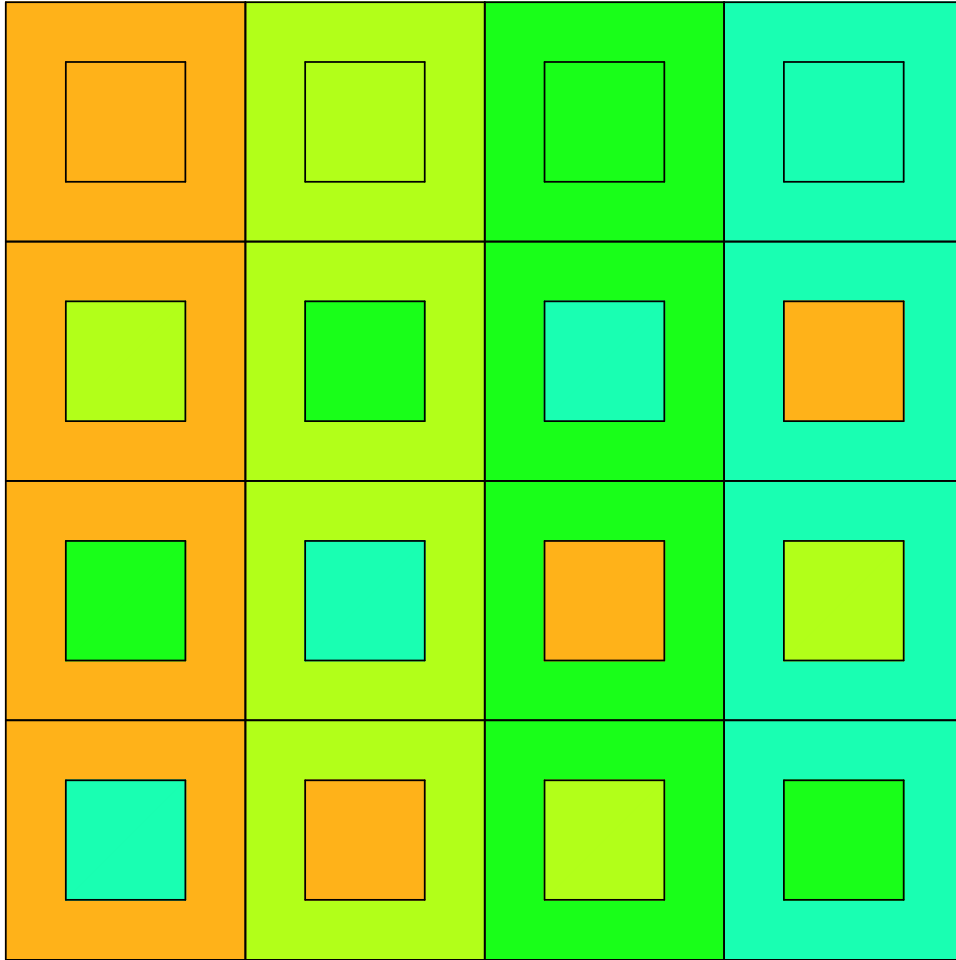
```
> plotmatrix(M,Z);
```



```
> S:=<<1 | 2 | 3 | 4> , <1| 2| 3 | 4> , <1 | 2 | 3| 4> , <1| 2 | 3  
| 4>>;
```

$$S := \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

```
> plotmatrix(M,S);
```



```

[ >
[ > islatin2 := proc(A)
  local dA,Z,S;
    dA := RowDimension(A);
    Z:= Matrix (dA,dA,(i,j)->i); Z hat in der ersten Zeile nur einsen,...usw.
    S:= Matrix (dA,dA,(i,j)->j); S hat in der ersten Spalte nur zweien,...usw.
    if orthogonal(A,Z) and orthogonal (A,S) then return true;
    else return false; end if;
  end proc:
[ > islatin2(Q1);
[                                     true
[ > with(linalg, randmatrix):
[ > st:= time():
[   for i from 1 to 1000 do
[     M:=RandomMatrix(4, 4):

```

```

    islatin(M):
end do:
time()-st;

```

1.612

```

> st:= time():
for i from 1 to 1000 do
    M:=RandomMatrix(4, 4):
    islatin2(M):
end do:
time()-st;

```

4.696

Hier wird deutlich, dass unsere erste Prozedur `islatin` effizienter arbeitet als die zweite.

Die Prozedur `permsquare` erzeugt eine n mal n Matrix deren erste Zeile die Zahlen von 1 bis n in aufsteigender Reihenfolge sind und die anderen Zeilen durch eine Permutation p bestimmt werden, indem die i -Zeile p hoch i darstellt.

```

permsquare := proc(L)
local n, M, f, i, j;
n:= nops(L);
M:= Matrix (n,n,[]);
f:= i-> L[i];
for i from 1 to n do
    for j from 1 to n do
        M[i,j]:= (f@@(i-1))(j);
    end do;
end do;
M;
end proc:

```

```

> L:=[2,4,1,3];

```

$L := [2, 4, 1, 3]$

```

> M4:=permsquare (L);

```

$$M4 := \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \\ 4 & 3 & 2 & 1 \\ 3 & 1 & 4 & 2 \end{bmatrix}$$

Die Methode `combinat [permute](L)` liefert eine Liste sämtlicher Permutationen von Elementen einer Liste L .

```

> L3:= [1,2,3];

```

$L3 := [1, 2, 3]$

```

> combinat[permute](L3);

```

$[[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]$

Die Prozedur `someLatinQ` liefert zu einer natürlichen Zahl n mit Hilfe der Prozedur `permsquare` einige (aber nicht zwangsläufig alle) lateinischen n -Quadrate

```

someLatinQ := proc(n)

```

```

local L,i,PermList,j,a,Output;
L:= [$1..n];
PermList:= combinat[permute](L);
Output:=NULL;
for a in PermList do
  if islatin(permsquare (a))=true then
Output:=Output,permsquare(a); end if;
end do;
Output;
end proc:

```

```
> someLatinQ(4);
```

```

[ [ 1 2 3 4 ] [ 1 2 3 4 ] [ 1 2 3 4 ] [ 1 2 3 4 ]
[ 2 3 4 1 ] [ 2 4 1 3 ] [ 3 1 4 2 ] [ 3 4 2 1 ]
[ 3 4 1 2 ] [ 4 3 2 1 ] [ 4 3 2 1 ] [ 2 1 4 3 ]
[ 4 1 2 3 ] [ 3 1 4 2 ] [ 2 4 1 3 ] [ 4 3 1 2 ]
[ 1 2 3 4 ] [ 1 2 3 4 ]
[ 4 1 2 3 ] [ 4 3 1 2 ]
[ 3 4 1 2 ] [ 2 1 4 3 ]
[ 2 3 4 1 ] [ 3 4 2 1 ]

```

dies sind aber nicht alle lateinischen 4-Quadrate, es gibt noch zwei andere, die sich aus der Kleinschen Vierergruppe ergeben: $V=\{\text{id}, (1\ 2)(3\ 4), (1\ 3)(2\ 4), (1\ 4)(2\ 3)\}$

Gesucht: orthogonale lateinische n-Quadrate mit $n = 3, 5$. Durch Permutieren der Zeilen 2 bis n enthält man möglicherweise weitere normierte lateinische Quadrate!

```
> someLatinQ(3);
```

```

[ [ 1 2 3 ] [ 1 2 3 ]
[ 2 3 1 ] [ 3 1 2 ]
[ 3 1 2 ] [ 2 3 1 ]

```

Die Prozedur ortholatin liefert zu einer festen Zahl n einige n-Quadrate, die orthogonal zueinander sind. Leider fehlerhaft....denn listentest(L,El) liefert false, obwohl El schon in L enthalten ist. Doch ich weiß nicht, wie ich dieses Problem beheben soll.

```
>
```

```

ortholatin := proc(n)
local L, El,i, j, k, l, m;
L:=someLatinQ(n); print(L);
for k from 1 to nops(L) do
for i from 2 to n-1 do
for j from i+1 to n do
  El:= RowOperation (L[k],[i,j]); print(listentest(L,El));
  if listentest(L,El)=false then L:=[op(L), El]; end if;
end do;
end do;
end do;

```

```

print (L);
for m from 1 to nops(L)-1 do
for l from m+1 to nops(L) do
    if islatin(L[m],L[l])=true then print([L[m],L[l]]); end if;
end do;
end do;
end proc:
> ortholatin(3);

```

```

      [[ 1 2 3]] [[ 1 2 3]]
      [[ 2 3 1]] [[ 3 1 2]]
      [[ 3 1 2]] [[ 2 3 1]]
listentest ([[ 1 2 3]] [[ 1 2 3]] [[ 1 2 3]])
            ([[ 2 3 1]] [[ 3 1 2]] [[ 3 1 2]])
            ([[ 3 1 2]] [[ 2 3 1]] [[ 2 3 1]])
listentest ([[ 1 2 3]] [[ 1 2 3]] [[ 1 2 3]] [[ 1 2 3]])
            ([[ 2 3 1]] [[ 3 1 2]] [[ 3 1 2]] [[ 2 3 1]])
            ([[ 3 1 2]] [[ 2 3 1]] [[ 2 3 1]] [[ 3 1 2]])
[[ 1 2 3]] [[ 1 2 3]] [[ 1 2 3]] [[ 1 2 3]]
[[ 2 3 1]] [[ 3 1 2]] [[ 3 1 2]] [[ 2 3 1]]
[[ 3 1 2]] [[ 2 3 1]] [[ 2 3 1]] [[ 3 1 2]]
[[ 1 2 3]] [[ 1 2 3]]
[[ 2 3 1]] [[ 3 1 2]]
[[ 3 1 2]] [[ 2 3 1]]
[[ 1 2 3]] [[ 1 2 3]]
[[ 2 3 1]] [[ 2 3 1]]
[[ 3 1 2]] [[ 3 1 2]]
[[ 1 2 3]] [[ 1 2 3]]
[[ 3 1 2]] [[ 3 1 2]]
[[ 2 3 1]] [[ 2 3 1]]

```

$$\left[\begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix} \right]$$

$$\left[\begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix} \right]$$

> L5:=[someLatinQ(5)];

$$L5 := \left[\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \\ 3 & 4 & 5 & 1 & 2 \\ 4 & 5 & 1 & 2 & 3 \\ 5 & 1 & 2 & 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 5 & 1 & 4 \\ 3 & 5 & 4 & 2 & 1 \\ 5 & 4 & 1 & 3 & 2 \\ 4 & 1 & 2 & 5 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 1 & 5 & 3 \\ 4 & 5 & 2 & 3 & 1 \\ 5 & 3 & 4 & 1 & 2 \\ 3 & 1 & 5 & 2 & 4 \end{bmatrix} \right]$$

$$\left[\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 4 & 5 & 3 & 1 \\ 4 & 3 & 1 & 5 & 2 \\ 3 & 5 & 2 & 1 & 4 \\ 5 & 1 & 4 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 1 & 3 & 4 \\ 5 & 4 & 2 & 1 & 3 \\ 4 & 3 & 5 & 2 & 1 \\ 3 & 1 & 4 & 5 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 5 & 4 & 1 & 3 \\ 5 & 3 & 1 & 2 & 4 \\ 3 & 4 & 2 & 5 & 1 \\ 4 & 1 & 5 & 3 & 2 \end{bmatrix} \right]$$

$$\left[\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 4 & 5 & 2 \\ 4 & 3 & 5 & 2 & 1 \\ 5 & 4 & 2 & 1 & 3 \\ 2 & 5 & 1 & 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 5 & 2 & 4 \\ 5 & 3 & 4 & 1 & 2 \\ 4 & 5 & 2 & 3 & 1 \\ 2 & 4 & 1 & 5 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 2 & 5 & 1 \\ 2 & 5 & 4 & 1 & 3 \\ 4 & 1 & 5 & 3 & 2 \\ 5 & 3 & 1 & 2 & 4 \end{bmatrix} \right]$$

$$\left[\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 1 & 2 \\ 5 & 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 & 1 \\ 4 & 5 & 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 2 & 1 & 4 \\ 2 & 4 & 5 & 3 & 1 \\ 5 & 1 & 4 & 2 & 3 \\ 4 & 3 & 1 & 5 & 2 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 4 & 2 & 1 \\ 4 & 1 & 2 & 5 & 3 \\ 2 & 3 & 5 & 1 & 4 \\ 5 & 4 & 1 & 3 & 2 \end{bmatrix} \right]$$

```

[ [ 1 2 3 4 5 ] [ 1 2 3 4 5 ] [ 1 2 3 4 5 ]
  [ 4 1 2 5 3 ] [ 4 1 5 3 2 ] [ 4 3 1 5 2 ]
  [ 5 4 1 3 2 ] [ 3 4 2 5 1 ] [ 5 1 4 2 3 ]
  [ 3 5 4 2 1 ] [ 5 3 1 2 4 ] [ 2 4 5 3 1 ]
  [ 2 3 5 1 4 ] [ 2 5 4 1 3 ] [ 3 5 2 1 4 ]
  [ 1 2 3 4 5 ] [ 1 2 3 4 5 ] [ 1 2 3 4 5 ]
  [ 4 3 5 2 1 ] [ 4 5 1 2 3 ] [ 4 5 2 3 1 ]
  [ 2 5 1 3 4 ] [ 2 3 4 5 1 ] [ 3 1 5 2 4 ]
  [ 3 1 4 5 2 ] [ 5 1 2 3 4 ] [ 2 4 1 5 3 ]
  [ 5 4 2 1 3 ] [ 3 4 5 1 2 ] [ 5 3 4 1 2 ]
  [ 1 2 3 4 5 ] [ 1 2 3 4 5 ] [ 1 2 3 4 5 ]
  [ 5 1 2 3 4 ] [ 5 1 4 2 3 ] [ 5 3 1 2 4 ]
  [ 4 5 1 2 3 ] [ 3 5 2 1 4 ] [ 4 1 5 3 2 ]
  [ 3 4 5 1 2 ] [ 4 3 1 5 2 ] [ 2 5 4 1 3 ]
  [ 2 3 4 5 1 ] [ 2 4 5 3 1 ] [ 3 4 2 5 1 ]
  [ 1 2 3 4 5 ] [ 1 2 3 4 5 ] [ 1 2 3 4 5 ]
  [ 5 3 4 1 2 ] [ 5 4 1 3 2 ] [ 5 4 2 1 3 ]
  [ 2 4 1 5 3 ] [ 2 3 5 1 4 ] [ 3 1 4 5 2 ]
  [ 3 1 5 2 4 ] [ 4 1 2 5 3 ] [ 2 5 1 3 4 ]
  [ 4 5 2 3 1 ] [ 3 5 4 2 1 ] [ 4 3 5 2 1 ]

```

```
[ > nops(L5);
```

```
24
```

```
[ >
```

```
[ >
```

```
> nops(
```

```
[
```

```
[ > L7);
```

```
720
```

```
[ Die Prozedur someLatinQ findet also 720 lateinische 7-Quadrate.
```

```
[ >
```

```
[ >
```

```
[ >
```