Laboratorium 8: Zastosowanie tranzytywnego domknięcia do partycjonowania czasu

Wariant pętli 3

```
for(i=1;i<=n;i++)
      for(j=3;j<=n;j++)
            a[i][j] = a[i][j-3];
```

Zadanie 1.
Dla wskazanej pętli za pomocą kalkulatora ISCC znaleźć relację zależności, R, oraz przestrzeń iteracji, LD.

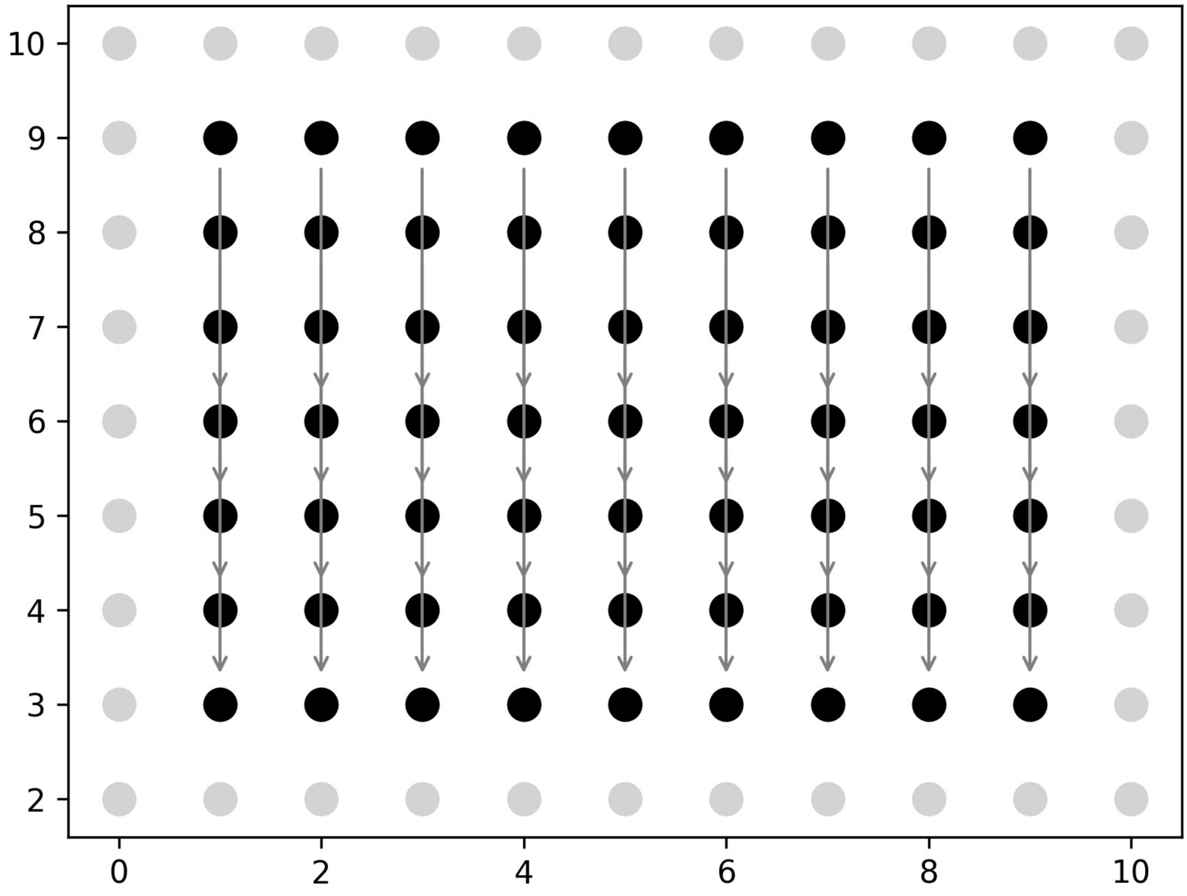
Relacja R:

```
[n] -> { [i, j] : 0 < i <= n and 3 <= j <= n }
```

Przestrzeń iteracji LD (Loop Domain):

```
[n] -> {[i, j] -> [i' = i, j' = 3 + j] : 0 < i <= n and 3 <= j <= -3 + n }
```

Zadanie 2.
Zrobić rysunek pokazujący zależności w przestrzeni 9 x 9. W tym celu trzeba zastosować operator scan (R*[n]->{:n=9}); który wygeneruje wszystkie zależności w przestrzeni 9 x 9.



Zadanie 3.
Utworzyć zbiór TILE dla rozmiaru kafelka 2x2

```
"TILE"
[n, it, jt] -> { [i, j] : it >= 0 and 2it <= -2 + n and jt >= 0 and 2jt <= -2 + n and 2it < i <=
2 + 2it and 2jt < j <= 2 + 2jt; [i, j = n] : 2jt = -1 + n and n > 0 and it >= 0 and 2it <= -2 + n
and 2it < i <= 2 + 2it; [i = n, j] : 2it = -1 + n and n > 0 and jt >= 0 and 2jt <= -2 + n and 2jt
< j <= 2 + 2jt; [i = n, j = n] : 2it = -1 + n and 2jt = -1 + n and n > 0 }
```

Zadanie 4.
Utworzyć zbiór TILE_LT

```
"TILE_LT"
[n, it, jt] -> { [i, j] : i <= 2 + 2it and j > 0 and 2*floor((1 + j)/2) <= n and ((jt >=
0 and 0 < i <= 2it and 2*floor((1 + i)/2) <= n) or (it >= 0 and 2it <= -2 + n and i > 2it
and j <= 2jt)); [i = n, j] : j > 0 and 2*floor((1 + j)/2) <= n and (((1 + n) mod 2 = 0
and 2it > n and jt >= 0) or (2it = -1 + n and j <= 2jt)); [i, j = n] : (1 + n) mod 2 = 0
and i <= 2 + 2it and ((jt >= 0 and 0 < i <= 2it and 2*floor((1 + i)/2) <= n) or (it >= 0
and 2it <= -2 + n and 2jt > n and i > 2it)); [i = n, j = n] : n > 0 and (((1 + n) mod 2 =
0 and 2it > n and jt >= 0) or (2it = -1 + n and 2jt > n)) }
```

Zadanie 5.
Utworzyć zbiór TILE_GT

```
"TILE_GT"
[n, it, jt] -> { [i, j] : it >= 0 and jt >= 0 and i > 2it and 2*floor((1 + j)/2) <= n and
((i >= 3 + 2it and j > 0 and 2*floor((1 + i)/2) <= n) or (2it <= -2 + n and i <= 2 + 2it
and j >= 3 + 2jt)); [i = n, j] : jt >= 0 and 2*floor((1 + j)/2) <= n and (((1 + n) mod 2
= 0 and it >= 0 and 2it <= -3 + n and j > 0) or (2it = -1 + n and j >= 3 + 2jt)); [i, j =
n] : (1 + n) mod 2 = 0 and it >= 0 and jt >= 0 and i > 2it and ((i >= 3 + 2it and
2*floor((1 + i)/2) <= n) or (2it <= -2 + n and 2jt <= -3 + n and i <= 2 + 2it)); [i = n,
j = n] : jt >= 0 and (((1 + n) mod 2 = 0 and it >= 0 and 2it <= -3 + n) or (2it = -1 + n
and 2jt <= -3 + n)) }
```

Zadanie 6.
Obliczyć relację R+

```
"RPLUS"
([n] -> { [i, j] -> [i' = i, j'] : (-j + j') mod 3 = 0 and 0 < i <= n and 3 <= j <= -3 +
n and j' >= 3 + j and 6 <= j' <= n }, True)
```

Zadanie 7.
Obliczyć zbiór TILE_ITR

```
"TILE_ITR"
[n, it, jt] -> { [i, j] : it >= 0 and 2it <= -2 + n and jt >= 0 and 2jt <= -2 + n and 2it
< i <= 2 + 2it and 2jt < j <= 2 + 2jt; [i, j = n] : 2jt = -1 + n and it >= 0 and 2it <=
-2 + n and 2it < i <= 2 + 2it; [i = n, j] : 2it = -1 + n and jt >= 0 and 2jt <= -2 + n
and 2jt < j <= 2 + 2jt; [i = n, j = n] : 2it = -1 + n and 2jt = -1 + n and n > 0 }
```

Zadanie 8.
Obliczyć zbiór TVLD_LT

```
"TVLD_LT"
[n, it, jt] -> {  }
```

Zadanie 9.

Obliczyć zbiór TILE_VLD

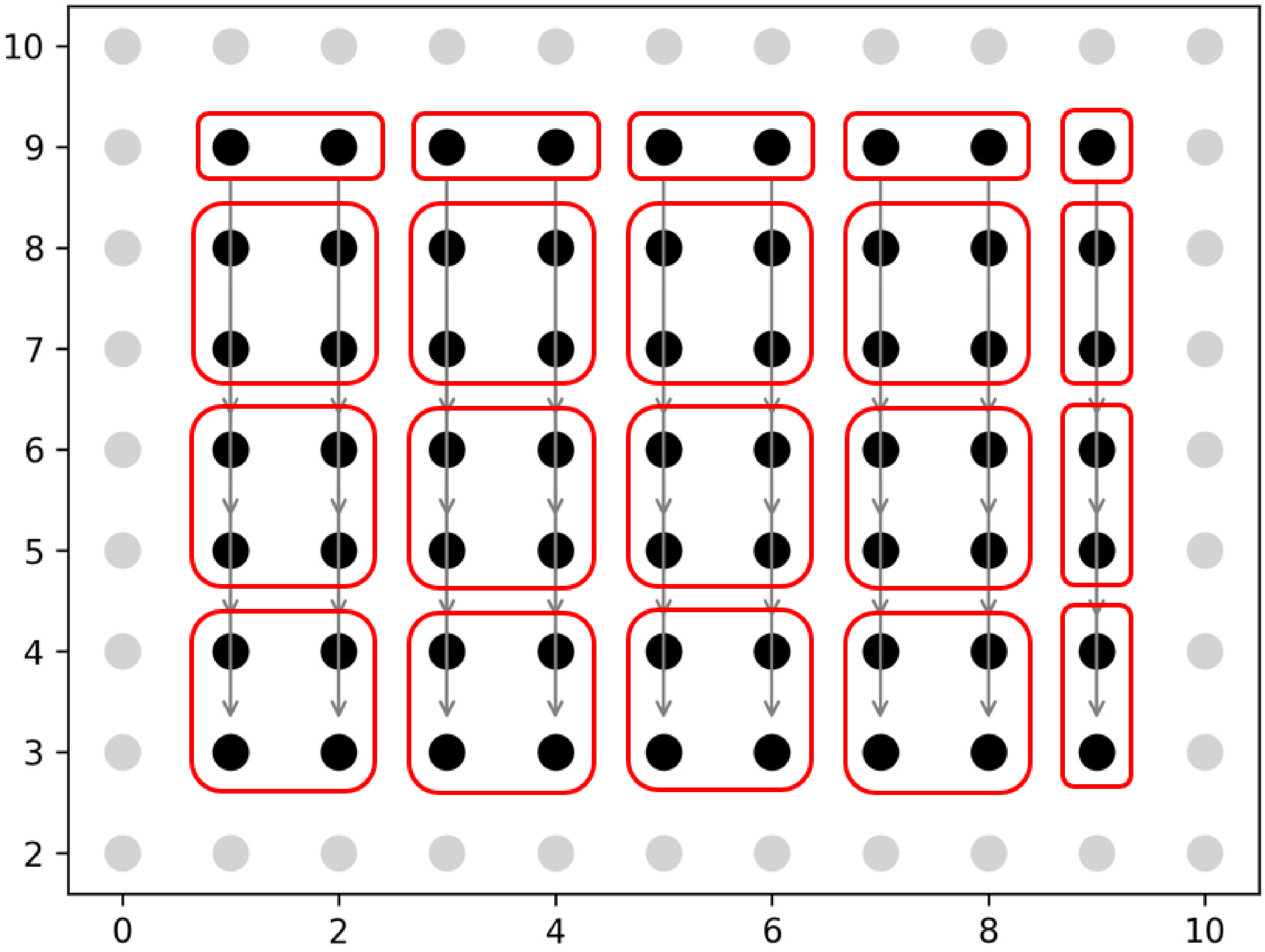
```
"TILE_VLD"
[n, it, jt] -> { [i, j] : it >= 0 and 2it <= -2 + n and jt >= 0 and 2jt <= -2 + n and 2it
< i <= 2 + 2it and 2jt < j <= 2 + 2jt; [i, j = n] : 2jt = -1 + n and it >= 0 and 2it <=
-2 + n and 2it < i <= 2 + 2it; [i = n, j] : 2it = -1 + n and jt >= 0 and 2jt <= -2 + n
and 2jt < j <= 2 + 2jt; [i = n, j = n] : 2it = -1 + n and 2jt = -1 + n and n > 0 }
```

Zadanie 10.

Sprawdzić zawartość kafelków za pomocą operatora scan kalkulatora iscc

```
"scan (TILE_VLD*[n]->{:n=6})"
```

```
[n, it, jt] -> {
[i = 9, j = 9] : n = 9 and it = 4 and jt = 4;
[i = 8, j = 9] : n = 9 and it = 3 and jt = 4;
[i = 7, j = 9] : n = 9 and it = 3 and jt = 4;
[i = 6, j = 9] : n = 9 and it = 2 and jt = 4;
[i = 5, j = 9] : n = 9 and it = 2 and jt = 4;
[i = 4, j = 9] : n = 9 and it = 1 and jt = 4;
[i = 3, j = 9] : n = 9 and it = 1 and jt = 4;
[i = 2, j = 9] : n = 9 and it = 0 and jt = 4;
[i = 1, j = 9] : n = 9 and it = 0 and jt = 4;
[i = 9, j = 8] : n = 9 and it = 4 and jt = 3;
[i = 8, j = 8] : n = 9 and it = 3 and jt = 3;
[i = 7, j = 8] : n = 9 and it = 3 and jt = 3;
[i = 6, j = 8] : n = 9 and it = 2 and jt = 3;
[i = 5, j = 8] : n = 9 and it = 2 and jt = 3;
[i = 4, j = 8] : n = 9 and it = 1 and jt = 3;
[i = 3, j = 8] : n = 9 and it = 1 and jt = 3;
[i = 2, j = 8] : n = 9 and it = 0 and jt = 3;
[i = 1, j = 8] : n = 9 and it = 0 and jt = 3;
[i = 9, j = 7] : n = 9 and it = 4 and jt = 3;
[i = 8, j = 7] : n = 9 and it = 3 and jt = 3;
[i = 7, j = 7] : n = 9 and it = 3 and jt = 3;
[i = 6, j = 7] : n = 9 and it = 2 and jt = 3;
[i = 5, j = 7] : n = 9 and it = 2 and jt = 3;
[i = 4, j = 7] : n = 9 and it = 1 and jt = 3;
[i = 3, j = 7] : n = 9 and it = 1 and jt = 3;
[i = 2, j = 7] : n = 9 and it = 0 and jt = 3;
[i = 1, j = 7] : n = 9 and it = 0 and jt = 3;
[i = 9, j = 6] : n = 9 and it = 4 and jt = 2;
[i = 8, j = 6] : n = 9 and it = 3 and jt = 2;
[i = 7, j = 6] : n = 9 and it = 3 and jt = 2;
[i = 6, j = 6] : n = 9 and it = 2 and jt = 2;
[i = 5, j = 6] : n = 9 and it = 2 and jt = 2;
[i = 4, j = 6] : n = 9 and it = 1 and jt = 2;
[i = 3, j = 6] : n = 9 and it = 1 and jt = 2;
[i = 2, j = 6] : n = 9 and it = 0 and jt = 2;
[i = 1, j = 6] : n = 9 and it = 0 and jt = 2;
[i = 9, j = 5] : n = 9 and it = 4 and jt = 2;
[i = 8, j = 5] : n = 9 and it = 3 and jt = 2;
[i = 7, j = 5] : n = 9 and it = 3 and jt = 2;
[i = 6, j = 5] : n = 9 and it = 2 and jt = 2;
[i = 5, j = 5] : n = 9 and it = 2 and jt = 2;
[i = 4, j = 5] : n = 9 and it = 1 and jt = 2;
[i = 3, j = 5] : n = 9 and it = 1 and jt = 2;
[i = 2, j = 5] : n = 9 and it = 0 and jt = 2;
[i = 1, j = 5] : n = 9 and it = 0 and jt = 2;
[i = 9, j = 4] : n = 9 and it = 4 and jt = 1;
[i = 8, j = 4] : n = 9 and it = 3 and jt = 1;
[i = 7, j = 4] : n = 9 and it = 3 and jt = 1;
[i = 6, j = 4] : n = 9 and it = 2 and jt = 1;
[i = 5, j = 4] : n = 9 and it = 2 and jt = 1;
[i = 4, j = 4] : n = 9 and it = 1 and jt = 1;
[i = 3, j = 4] : n = 9 and it = 1 and jt = 1;
[i = 2, j = 4] : n = 9 and it = 0 and jt = 1;
[i = 1, j = 4] : n = 9 and it = 0 and jt = 1;
[i = 9, j = 3] : n = 9 and it = 4 and jt = 1;
[i = 8, j = 3] : n = 9 and it = 3 and jt = 1;
[i = 7, j = 3] : n = 9 and it = 3 and jt = 1;
[i = 6, j = 3] : n = 9 and it = 2 and jt = 1;
[i = 5, j = 3] : n = 9 and it = 2 and jt = 1;
[i = 4, j = 3] : n = 9 and it = 1 and jt = 1;
[i = 3, j = 3] : n = 9 and it = 1 and jt = 1;
[i = 2, j = 3] : n = 9 and it = 0 and jt = 1;
[i = 1, j = 3] : n = 9 and it = 0 and jt = 1;
[i = 9, j = 2] : n = 9 and it = 4 and jt = 0;
[i = 8, j = 2] : n = 9 and it = 3 and jt = 0;
[i = 7, j = 2] : n = 9 and it = 3 and jt = 0;
[i = 6, j = 2] : n = 9 and it = 2 and jt = 0;
[i = 5, j = 2] : n = 9 and it = 2 and jt = 0;
[i = 4, j = 2] : n = 9 and it = 1 and jt = 0;
[i = 3, j = 2] : n = 9 and it = 1 and jt = 0;
[i = 2, j = 2] : n = 9 and it = 0 and jt = 0;
[i = 1, j = 2] : n = 9 and it = 0 and jt = 0;
[i = 9, j = 1] : n = 9 and it = 4 and jt = 0;
[i = 8, j = 1] : n = 9 and it = 3 and jt = 0;
[i = 7, j = 1] : n = 9 and it = 3 and jt = 0;
[i = 6, j = 1] : n = 9 and it = 2 and jt = 0;
[i = 5, j = 1] : n = 9 and it = 2 and jt = 0;
[i = 4, j = 1] : n = 9 and it = 1 and jt = 0;
[i = 3, j = 1] : n = 9 and it = 1 and jt = 0;
[i = 2, j = 1] : n = 9 and it = 0 and jt = 0;
[i = 1, j = 1] : n = 9 and it = 0 and jt = 0
}
```



Zadanie 11.

Utworzyć zbiór TILE_VLD_EXT

```
TILE_VLD_EXT:=[n] -> {
    [it, jt, i, j] :
        it >= 0 and
        2it <= -2 + n and
        jt >= 0 and
        2jt <= -2 + n and
        2it < i <= 2 + 2it and
        2jt < j <= 2 + 2jt;

    [it, jt, i, j = n] :
        2jt = -1 + n and
        it >= 0 and
        2it <= -2 + n and
        2it < i <= 2 + 2it;
    [it, jt, i = n, j] :
        2it = -1 + n and
        jt >= 0 and
        2jt <= -2 + n and
        2jt < j <= 2 + 2jt;

    [it, jt, i = n, j = n] :
        2it = -1 + n and
        2jt = -1 + n and
        n > 0;
};
```

Zadanie 12.
Przekształcić zbiór TILE_VLD_EXT na relacje CODE

```
CODE:=identity TILE_VLD_EXT;
```

Lub w przypadku gdy n=6:

```
CODE:=identity (TILE_VLD_EXT * [n]->{:n=6});
```

Zadanie 13.
Wygenerować kod za pomocą operatora codegen

W przypadku gdy n nie jest określone:

```
for (int c0 = 0; c0 < floord(n + 1, 2); c0 += 1)
  for (int c1 = 0; c1 < (n + 1) / 2; c1 += 1)
    for (int c2 = 2 * c0 + 1; c2 <= min(n, 2 * c0 + 2); c2 += 1) {
      if (n >= 2 * c0 + 2) {
        for (int c3 = 2 * c1 + 3; c3 <= min(n, 2 * c1 + 5); c3 += 1)
          (c0, c1, c2, c3);
      } else if (n >= 2 * c1 + 3) {
        for (int c3 = 2 * c1 + 1; c3 <= 2 * c1 + 2; c3 += 1)
          ((n - 1) / 2, c1, n, c3);
      } else {
        ((n - 1) / 2, (n - 1) / 2, n, n);
      }
    }
```

W przypadku gdy n=6:

```
if (n == 6)
  for (int c0 = 0; c0 <= 2; c0 += 1)
    for (int c1 = 0; c1 <= 2; c1 += 1)
      for (int c2 = 2 * c0 + 1; c2 <= 2 * c0 + 2; c2 += 1)
        for (int c3 = 2 * c1 + 1; c3 <= 2 * c1 + 2; c3 += 1)
          (c0, c1, c2, c3);
```

Kod kompilowalny

```
for (int c0 = 0; c0 < floord(n + 1, 2); c0 += 1)
     for (int c1 = 0; c1 < (n + 1) / 2; c1 += 1)
```

```
        for (int c2 = 2 * c0 + 1; c2 <= min(n, 2 * c0 + 2); c2 += 1) {
            if (n >= 2 * c0 + 2) {
                for (int c3 = 2 * c1 + 3; c3 <= min(n, 2 * c1 + 5); c3 += 1)
                    aGenerated[c2][c3] = aGenerated[c2][c3-3];
            }
            else if (n >= 2 * c1 + 3) {
                for (int c3 = 2 * c1 + 1; c3 <= 2 * c1 + 2; c3 += 1)
                    aGenerated[c2][c3] = aGenerated[c2][c3-3];
            }
            else {
                aGenerated[n][n] = aGenerated[n][n-3];
            }
        }
```

Zadanie 14.
Zastosować program porównujący wyniki obliczeń do sprawdzania poprawności kodu docelowego w przestrzeni 6x6.

```
# gcc -fopenmp 2-joined.c -lm && ./a.out
Initial code result:
00 01 02 03 04 05 06
00 01 02 00 01 02 00
00 01 02 00 01 02 00
00 01 02 00 01 02 00
00 01 02 00 01 02 00
00 01 02 00 01 02 00
00 01 02 00 01 02 00

Generated code result:
00 01 02 03 04 05 06
00 01 02 00 01 02 00
00 01 02 00 01 02 00
00 01 02 00 01 02 00
00 01 02 00 01 02 00
00 01 02 00 01 02 00
00 01 02 00 01 02 00
```

Załączniki.

Skrypt implementujący zadania.

```
##krok 1: relacja zaleznosci, R, oraz przestrzen iteracji,LD:
LD := [n] -> { [i, j] : 0 < i <= n and 3 <= j <= n }
print "LD"; LD;

### relacja zaleznosci
R := [n] -> { [i, j] -> [i' = i, j' = 3 + j] : 0 < i <= n and 3 <= j <= -3 + n };
print "R"; R;

#Tworzenie zbioru TILE:
## szerokosc kafelka = 2
TILE:=[n, it, jt]->{
    [i,j]:
        2it + 1 <= i <= min(2 * (it + 1), n) and
        2jt + 1 <= j <= min(2 * (jt + 1), n) and
        it,jt >= 0
};
print "TILE"; TILE;

#TILE_LT obliczamy nastepujaco:
TILE_LT:=[n, it, jt]->{[i, j]: exists it',jt': (it' < it or it' = it and jt' < jt)
    and 2it' + 1 <= i <= min(2 * (it' + 1), n)
    and 2jt' + 1 <= j <= min(2 * (jt' + 1), n)
    and it, it', jt, jt' >= 0
};
print "TILE_LT"; TILE_LT;

#TILE_GT obliczamy nastepujaco:
TILE_GT:=[n,it,jt]->{[i, j]: exists it', jt': (it' > it or it' = it and jt' > jt)
    and 2it' + 1 <= i <= min(2 * (it' + 1),n)
    and 2jt' + 1 <= j <= min(2*(jt' + 1),n)
    and it, it', jt, jt' >= 0
};
print "TILE_GT"; TILE_GT;

##obliczenie relacji R+
RPLUS:=R^+;
print "RPLUS"; RPLUS;

##Obliczenie zbioru TILE_ITR
TILE_ITR:= TILE  - RPLUS(TILE_GT);
print "TILE_ITR"; TILE_ITR;

##obliczenie zbioru TVLD_LT
TVLD_LT:= (RPLUS (TILE_ITR) * TILE_LT) -RPLUS(TILE_GT);
print "TVLD_LT"; TVLD_LT;

##obliczenie zbioru TILE_VLD
TILE_VLD:= TILE_ITR  +  TVLD_LT;
print "TILE_VLD"; TILE_VLD;

##celem sprawdzenia zawartosci kafelkow mozemy skorzystac z #operatora scan:
#print "scan (TILE_VLD*[n]->{:n=6})";
scan (TILE_VLD*[n]->{:n=9});
print "-----------------------------";

##tworzenie zbioru TILE_VLD_EXT
# parametry it,jt trzeba przenisc na pierwsze pozycje kazdej krotki zbioru TILE_VLD:

TILE_VLD_EXT:=[n] -> {
    [it, jt, i, j] :
        it >= 0 and
        2it <= -2 + n and
        jt >= 0 and
        2jt <= -2 + n and
        2it < i <= 2 + 2it and
        2jt < j <= 2 + 2jt;
    [it, jt, i, j = n] :
        2jt = -1 + n and
        it >= 0 and
        2it <= -2 + n and
        2it < i <= 2 + 2it;

    [it, jt, i = n, j] :
        2it = -1 + n and
        jt >= 0 and
        2jt <= -2 + n and
        2jt < j <= 2 + 2jt;

    [it, jt, i = n, j = n] :
        2it = -1 + n and
        2jt = -1 + n and
        n > 0;
};


##konwertujemy zbior TILE_VLD_EXT na relacje CODE"

# CODE:=identity (TILE_VLD_EXT * [n]->{:n=6});
CODE:=identity TILE_VLD_EXT;
codegen CODE;
```