Laboratorium 5. Cel: Zastosowanie szeregowania afinicznego do implementacji techniki blokowania pętli (patrz wykład 10), 2 godziny.

Zadania:

1. Dla wskazanej pętli za pomocą kalkulatora ISCC znaleźć relację zależności, R, przestrzeń iteracji, LD, oraz zrobić rysunek grafu zależności w przestrzeni 6 x 6. W tym celu trzeba zastosować operator scan (R*[n]->{:n=6}); który wygeneruje wszystkie zależności w przestrzeni 6 x 6, pierwsza krotka wskazuje początek zależności (strzałki), druga krotka – koniec zależności (strzałki).
   !!!Uwaga: dla niektórych pętli w przestrzeni 6x6 zależności mogą nie istnieć, w takim przypadku należy rozszerzyć przestrzeń do rozmiaru 12x12.
2. Za pomocą operatora kalkulatora ISCC: *IslSchedule := schedule LD respecting R minimizing R* znaleźć szeregowanie afiniczne w postaci drzewa.
3. Za pomocą operatora kalkulatora ISCC: *map* przekonwertować szeregowanie afiniczne w postaci drzewa na szeregowanie w postaci relacji.
4. Utworzyć szeregowanie, które pozwala na implementację techniki fali frontowej na poziomie iteracji (wave-fronting).
5. Utworzyć szeregowanie, które pozwala na implementację techniki kafelkowania (tiling) z rozmiarem kafelka 2x2, sekwencyjny sposób wykonywania kafelków.
6. Stosując uzyskane w punkcie 5 szeregowanie za pomocą operatora *scan* znaleźć wszystkie kafelki w przestrzeni 6x6 (12z12) i zaznaczyć je na rysunku utworzonego w p.1.

   Żeby zaznaczyć kafelki wynikowe, trzeba posortować wyniki, które zwraca operator *scan*, tak, żeby było jasne które punkty należą do tego samego kafelka.
   Ten sam kafelek zawiera punkty z tym samym identyfikatorem, który jest określony przez wartości iteratorów it oraz jt.

7. Wygenerować pseudokod sekwencyjny i odpowiedni kod kompilowany (sekwencyjny) implementujący kafelkowanie.
8. Zastosować program porównujący wyniki obliczeń (zadanie 7, L2) do sprawdzania poprawności kodu docelowego wygenerowanego w p.6 w przestrzeni 6x6.
9. Utworzyć szeregowanie, które pozwala na implementację techniki wave-fronting na poziomie kafli (tiles) z rozmiarem kafelka 2x2, równoległy sposób wykonywania kafelkow.
10. Stosując uzyskane w punkcie 9 szeregowanie za pomocą operatora *scan* znaleźć wszystkie partycje czasu na poziomie kafelków w przestrzeni 6x6 (12z12) i zaznaczyć je na rysunku utworzonego w p.1.
    Czasem wykonania kafelka jest wartość iteratora it.
11. Wygenerować pseudokod i równoległy kod kompilowalny implementujący kafelkowanie.
12. Zastosować program porównujący wyniki obliczeń (zadanie 7, L2) do sprawdzania poprawności kodu docelowego w przestrzeni 6x6.
13. Opracować sprawozdanie.

Patrz skrypt skrypt L5 pokazujący dla przykładowej pętli realizację poszczególnych zadań wyżej.

Warianty pętli:

1.
```
for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
    a[i][j] = a[i][j-1];
```
2.
```
for(i=1;i<=n;i++)
  for(j=2;j<=n;j++)
    a[i][j] = a[i][j-2];
```
3.
```
for(i=1;i<=n;i++)
  for(j=2;j<=n;j++)
    a[i][j] = a[i][j-2];
```
4.
```
for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
    a[i][j] = a[i-1][j-1];
```
5.
```
for(i=2;i<=n;i++)
  for(j=2;j<=n;j++)
    a[i][j] = a[i-2][j-1];
```
6.
```
for(i=2;i<=n;i++)
  for(j=2;j<=n;j++)
    a[i][j] = a[i-2][j-2];
```
7.
```
for(i=2;i<=n;i++)
  for(j=2;j<=n;j++)
    a[i][j] = a[i-2][j+2];
```
8.
```
for(i=1;i<=n;i++)
  for(j=0;j<=n;j++)
    a[i][j] = a[i-1][j+2];
```
9.
```
for(i=1;i<=n;i++)
  for(j=0;j<=n;j++)
    a[i][j] = a[i-1][j+1];
```
10.
```
for(i=1;i<=n;i++)
  for(j=0;j<=n;j++)
    a[i][j] = a[i+3][j+4];
```
11.
```
for(i=1;i<=n;i++)
  for(j=4;j<=n;j++)
    a[i][j] = a[i+3][j-4];
```
12.
```
for(i=1;i<=n;i++)
  for(j=4;j<=n;j++)
    a[i][j] = a[i+4][j-4];
```
13.

```
for(i=1;i<=n;i++)
  for(j=4;j<=n;j++)
    a[i][j] = a[i+5][j-4];
```

14.

```
for(i=1;i<=n;i++)
  for(j=4;j<=n;j++)
    a[i][j] = a[i+5][j-3];
```

15.

```
for(i=1;i<=n;i++)
  for(j=4;j<=n;j++)
    a[i][j] = a[i+5][j-2];
```

16.

```
for(i=1;i<=n;i++)
  for(j=4;j<=n;j++)
    a[i][j] = a[i+5][j-1];
```

Sprawozdanie powinno zawierać: pętlę, skrypt implementujący zadania oraz wyniki wszystkich kroków.