Laboratorium 4. Cel: Zastosowanie szeregowania afinicznego do znalezienia partycjonowania czasu (równoległości z synchronizacją - 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 (wave-fronting) na poziomie iteracji, patrz skrypt L4.
5. Stosując uzyskane w punkcie 4 szeregowanie za pomocą operatora *scan* znaleźć wszystkie partycje czasu dla przestrzeni 6x6 (12x12) i nanieść uzyskane partycje na rysunek utworzony w p.1.
6. Wygenerować pseudokod i kod kompilowalny implementujący technikę fali frontowej.
7. Zastosować program porównujący wyniki obliczeń (zadanie 7, L2) do sprawdzenia poprawności kodu docelowego w przestrzeni 6x6 (12z12).
8. Wygenerować kod reprezentujący kompilowalną pętlę całkowicie wymienną (patrz skrypt L4).
9. Sprawdzić, że jest to pętla całkowicie wymienna poprzez obliczenie relacji zależności dla pętli wygenerowanej w p. 8 i zastosowanie operatora *deltas* do tej relacji. Wszystkie elementy uzyskanego wektora dystansu powinny być nieujemne.
10. Opracować sprawozdanie.

Patrz skrypt skrypt_L4 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] +a[i+1][j];
```
2.
```
for(i=1;i<=n;i++)
  for(j=2;j<=n;j++)
    a[i][j] = a[i][j-2] +a[i+1][j];
```
3.
```
for(i=1;i<=n;i++)
  for(j=3;j<=n;j++)
    a[i][j] = a[i][j-3] +a[i+1][j];
```

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

14.
```
for(i=1;i<=n;i++)
  for(j=4;j<=n;j++)
    a[i][j] = a[i+5][j-4] +a[i+1][j+1];
```

15.

```
for(i=1;i<=n;i++)
  for(j=0;j<=n;j++)
    a[i][j] = a[i+3][j+4] +a[i+1][j]+1;
```

16.

```
for(i=1;i<=n;i++)
  for(j=0;j<=n;j++)
    a[i][j] = a[i-1][j+1] +a[i+1][j];
```

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