Laboratorium 6. Cel: Zastosowanie tranzytywnego domknięcia do znalezienia równoległości pozbawionej synchronizacji (patrz wykład 11), 2 godziny

Zadania:

1. Dla wskazanej pętli za pomocą kalkulatora ISCC znaleźć relację zależności, R, oraz przestrzeń iteracji, LD.
2. Zrobić rysunek pokazujący 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.
3. Znaleźć początki krańcowe reprezentowane przez zbiór UDS.
4. Obliczyć relację R_USC.
5. Określić jaka jest topologia grafu zależności.
6. Znaleźć punkty reprezentatywne niezależnych fragmentów grafu, czyli zbiór REPR.
7. Znaleźć zbiór, SLICES, zawierający wszystkie iteracje należące do niezależnego fragmentu z danym punktem reprezentatywnym, I.
8. Stosując zbiór SLICES za pomocą operatora *scan* znaleźć wszystkie niezależne fragmenty kodu i zaznaczyć je na rysunku utworzonym w p. 2 (rysunek z zależnościami) w przestrzeni 6x6 (12x12).
   Żeby zrobić rysunek trzeba wziąć pod uwagę, że pierwsza para zmiennych w krotce zbioru SLICES określa punkt reprezentatywny niezależnego fragmentu kodu (jego identyfikator) natomiast druga para reprezentuje iterację, która należy do fragmentu z identyfikatorem określonym przez wartości zmiennych pierwszej pary.
   Do tego samego fragmentu należą punkty, które mają ten sam identyfikator.

9. Wygenerować pseudokod.
10. Przetransformować pseudokod na kod kompilowany w OpenMP.
11. Zastosować program opracowany w (p.7, L2) do sprawdzenia poprawności kodu docelowego w przestrzeni 6x6 (12x12).
12. Opracować sprawozdanie.

Patrz skrypt skrypt_L6 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=3;j<=n;j++)
    a[i][j] = a[i][j-3];
```

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=1;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];
```

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