**Zadanie 2. Cel**: Znajdowanie zależności z wykorzystaniem PET i kalkulatora ISCC; generowanie kodu; implementacja postprocesora, opracowanie programu do porównania wyników produkowanych przez program źródłowy i docelowy, 2 godziny.

**Zadania:**

1. Dla wskazanej pętli za pomocą PET i kalkulatora ISCC znaleźć relację zależności, R. Uwaga!!! Kalkulator iscc dostępny on-line nie korzysta z PET.
2. Dla wskazanego szeregowania wygenerować pseudokod implementujący wave-fronting (patrz skrypt_L2).
3. Opracować kod (na wejściu preprocesora), który przekształca pseudokod na kod kompilowalny.
4. Za pomocą preprocesora uzyskać kod docelowy (kompilowalny)
5. Opracować aplikację do porównania wyników produkowanych przez program źródłowy i docelowy (będzie potrzebna dla każdego kolejnego laboratorium).
6. Opracować sprawozdanie.

   Wywołanie iscc: iscc < my.iscc, gdzie my.iscc jest to dowolny skrypt iscc
   Patrz skrypt „skrypt_L2", zawierający dodatkowe wskazówki.

**Warianty pętli:**

1.
```
for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
    a[i][j] = a[2i][j-1];
```
2.
```
for(i=1;i<=n;i++)
  for(j=2;j<=n;j++)
    a[i][j] = a[2i][j-2];
```
3.
```
for(i=1;i<=n;i++)
  for(j=2;j<=n;j++)
    a[i][j] = a[3i][j-2];
```
4.
```
for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
    a[i][j] = a[2i-1][j-1];
```
5.
```
for(i=2;i<=n;i++)
  for(j=1;j<=n;j++)
    a[i][j] = a[2i-2][j-1];
```
6.
```
for(i=2;i<=n;i++)
  for(j=2;j<=n;j++)
    a[i][j] = a[2i-2][j-2];
```
7.
```
for(i=2;i<=n;i++)
  for(j=2;j<=n;j++)
```

```
    a[i][j] = a[2i-2][j+2];
```
8.
```
for(i=1;i<=n;i++)
  for(j=0;j<=n;j++)
    a[i][j] = a[2i-1][j+2];
```

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

14.
```
for(i=1;i<=n;i++)
  for(j=5;j<=n;j++)
    a[i][j] = a[2i+5][j-4];
```

15.
```
for(i=1;i<=n;i++)
  for(j=6;j<=n;j++)
    a[i][j] = a[2i+5][j-4];
```

16.
```
for(i=1;i<=n;i++)
  for(j=6;j<=n;j++)
    a[i][j] = a[2i+5][j-5];
```

Warianty szeregowania:

1. SCHED:=[n]->{[i,j]->[2i+j][j]:  1<=i,j<=n};
2. SCHED:=[n]->{[i,j]->[2i+2j][j]:  1<=i,j<=n};
3. SCHED:=[n]->{[i,j]->[i+2j][j]:  1<=i,j<=n};
4. SCHED:=[n]->{[i,j]->[i+3j][j]:  1<=i,j<=n};

5. SCHED:=[n]->{[i,j]->[2i+3j][j]: 1<=i,j<=n};
6. SCHED:=[n]->{[i,j]->[i+4j][j]: 1<=i,j<=n};
7. SCHED:=[n]->{[i,j]->[2i+4j][j]: 1<=i,j<=n};
8. SCHED:=[n]->{[i,j]->[i+5j][j]: 1<=i,j<=n};
9. SCHED:=[n]->{[i,j]->[2i+5j][j]: 1<=i,j<=n};
10. SCHED:=[n]->{[i,j]->[2i-j][j]: 1<=i,j<=n};
11. SCHED:=[n]->{[i,j]->[3i-j][j]: 1<=i,j<=n};
12. SCHED:=[n]->{[i,j]->[2i+6j][j]: 1<=i,j<=n};
13. SCHED:=[n]->{[i,j]->[3i+6j][j]: 1<=i,j<=n};
14. SCHED:=[n]->{[i,j]->[3i+7j][j]: 1<=i,j<=n};
15. SCHED:=[n]->{[i,j]->[4i+7j][j]: 1<=i,j<=n};
16. SCHED:=[n]->{[i,j]->[3i+8j][j]: 1<=i,j<=n};

**Sprawozdanie powinno zawierać:** wskazaną pętlę, plik z poprawnym kodem zawierającym pętlę, skrypt do znalezienia zależności, relację zależności, wskazane szeregowanie, pseudokod przebierający iteracje pętli według wskazanego szeregowania, kod do przekształcenia pseudokodu na kod równoległy w OpenMP, kod w OpenMP, program do porównania wyników produkowanych przez program oryginalny i docelowy.