Cechy CSP (omówione na zajęciach):

- podstawą sa komunikujące się procesy, każdy z własną pamięcią
- komunikacja poprzez symetryczne spotkania (w Kotlinie realizowane poprzez kanały, w pseudokodzie poprzez operatory "?" i "!" )
- quards, instrukcja alternatywy, instrukcja wykonania równoległego

Wyjaśnienie notacji pseudokodu dla CSP jest dołączone do tego zadania (plik CSP_notation_expolained.pdf).

Będziemy używać adaptacji formalizmu CSP do języka Kotlin.
Najważniejsze powiązania:
procesy CSP = corutyny https://kotlinlang.org/docs/coroutines-basics.html
symetryczne spotkania = kanały bez bufora typu *rendezvous* https://kotlinlang.org/docs/channels.html
instrukcja alternatywy = wyrażenie select https://kotlinlang.org/docs/select-expression.html

Ciekawy przykład https://proandroiddev.com/kotlin-coroutines-channels-csp-android-db441400965f

**Zadanie 1 (2 pkt)**
**Zaimplementuj rozwiązanie problemu producenta i konsumenta z pośrednikiem. Pośredników jest wielu, ale do przekazania produktu należy wykorzystać jednego, losowo wybranego (każde połączenie producent-pośrednik należy zrealizować na osobnym kanale i użyć wyrażenia select )**

Pseudokod w notacji CSP

```
// Producent i konsument z wybranym pośrednikiem
// N - liczba posredników
[PRODUCER:: p: porcja;
*[true -> produkuj(p);
[(i:0..N-1) POSREDNIK(i)?JESZCZE() -> POSREDNIK(i)!p]
]
||POSREDNIK(i:0..N-1):: p: porcja;
*[true -> PRODUCER!JESZCZE() ;
[PRODUCER?p -> CONSUMER!p]
]
||CONSUMER:: p: porcja;
*[(i:0..N-1) POSREDNIK(i)?p -> konsumuj(p)]
```

*Zadanie 2 (2 pkt)*
**Zaimplementuj w rozwiązanie problemu taśmy produkcyjnej,**
**gdzie producent przekazuje dane do procesu przetwarzającego 0, ten podaje produkt do procesu przetwarzającego 1 itd, aż do konsumenta**

Pseudokod

```
// Producent i konsumer  oraz procesy przetwarzające
// N - liczba przetwarzaczy;
[PRODUCER:: p: porcja;
*[true -> produkuj(p); PRZETWARZACZ(0)!p]
||PRZETWARZACZ(i:O..N-l):: p: porcja;
*[true -> [i = 0 -> PRODUCER?p
[]i <> 0 -> PRZETWARZACZ(i-l)?p];
[i = N-l -> CONSUMER!p
[]i <> N-l -> PRZETWARZACZ(i+l)!p]
]
|| CONSUMER:: p: porcja;
*[PRZETWARZACZ(N-l)?p -> konsumuj(p)]
]
```