Zadanie 1.

1) Napisać program współbieżny symulujący działanie m producentów i n konsumentów komunikujących się przez ograniczony bufor cykliczny.
2) Działanie wątku producenta polega na wielokrotnym wykonywaniu po sobie ciągów instrukcji odpowiadających tzw. *produkcji danych* oraz synchronizowanego wstawienia ich do buforu. Produkcja danych polegać ma na wstrzymaniu wątku przez losowy czas w przedziale <1, 10> milisekund oraz wylosowaniu liczby całkowitej z przedziału <0, 99>. Dana jest typu string i ma następującą postać:

   Dana=[*id producenta*, *nr powt.*, pozycja w buforze, *wartość*]

   `Dana=[P-1, 100, 3, 88]`

3) Działanie wątku konsumenta polega na wielokrotnym wykonywaniu po sobie synchronizowanego pobrania danych z buforu oraz tzw. *konsumpcji danych*. Konsumpcja danych polegać ma na wstrzymaniu wątku przez losowy czas w przedziale <2, 12> milisekund oraz wypisania stosownego komunikatu.
4) Komunikat powinien mieć postać:

   [id konsumenta, nr powt., pozycja w buforze] :: Dana=[*id producenta*, *nr powt.*, pozycja w buforze, *wartość*]

   `[K-2, 33, 3] :: Dana=[P-1, 100, 3, 88]`

5) W celu synchronizacji procesów wykorzystać obiekty klasy `Semaphore`.
6) Przerwać wykonywanie wątków potomnych po zadanym czasie np. 10s. (wykorzystać metodę `interrupt()` i `join()`).
7) Działanie programu zademonstrować dla m=4, n=5.

Zadanie 2.

1) Napisać program współbieżny symulujący działanie m czytelników i n pisarzy korzystających ze współdzielonej czytelni.
2) Działanie wątku czytelnika oraz pisarza polega na wielokrotnym wykonywaniu po sobie ciągów instrukcji odpowiadających tzw. *sprawom własnym* oraz synchronizowanemu korzystaniu z czytelni. Sprawy własne polegać maja na wstrzymaniu wątku przez losowy czas w przedziale <5, 15> milisekund.
3) Synchronizowane korzystanie z czytelni dla wątków czytelników i pisarzy objawia się wstrzymaniem wątku przez losowy czas w przedziale <1, 5> milisekund oraz wypisaniem stosownych komunikatów.
4) Wątki czytelników wypisują komunikat przed wejściem i po wyjściu do/z czytelni. Komunikat powinien mieć postać (początkowe symbole oznaczają: **„>>>”** – przed, **„<<<”** – po):

   >>> [id wątku, nr powt.] :: [licz_czyt=2, licz_czyt_pocz=1, licz_pis=0, licz_pis_pocz=1]

   `>>> [C-2, 33] :: [licz_czyt=2, licz_czyt_pocz=1, licz_pis=0, licz_pis_pocz=1]`

5) Wątki pisarzy wypisują komunikat przed wejściem i po wyjściu do/z czytelni. Komunikat powinien mieć postać (początkowe symbole oznaczają: **„==>”** – przed, **„<==”** – po):

   ==> [id wątku, nr powt.] :: [licz_czyt=2, licz_czyt_pocz=1, licz_pis=0, licz_pis_pocz=1]

   `==> [P-2, 33] :: [licz_czyt=2, licz_czyt_pocz=1, licz_pis=0, licz_pis_pocz=1]`

6) W celu synchronizacji procesów wykorzystać obiekty klasy `Semaphore`.
7) Przerwać wykonywanie wątków potomnych po zadanym czasie np. 10s. (wykorzystać metodę `interrupt()` i `join()`).
8) Działanie programu zademonstrować dla m=4, n=2.

Zadanie 3.

1) Napisać program współbieżny symulujący działanie m czytelników i n pisarzy korzystających ze współdzielonej czytelni o pojemności K < m.
2) Działanie wątku czytelnika oraz pisarza polega na wielokrotnym wykonywaniu po sobie ciągów instrukcji odpowiadających tzw. *sprawom własnym* oraz synchronizowanemu korzystaniu z czytelni. Sprawy własne polegać maja na wstrzymaniu wątku przez losowy czas w przedziale <5, 15> milisekund.
3) Synchronizowane korzystanie z czytelni dla wątków czytelników i pisarzy objawia się wstrzymaniem wątku przez losowy czas w przedziale <1, 5> milisekund oraz wypisaniem stosownych komunikatów podczas rozpoczęcia i zakończenia korzystania z czytelni.
4) Wątki czytelników wypisują komunikat przed wejściem i po wyjściu do/z czytelni. Komunikat powinien mieć postać (początkowe symbole oznaczają: „>>>" – przed, „<<<" – po):

   >>> [id wątku, nr powt.] :: [licz_czyt=2, licz_pis=0]

   `>>> [C-2, 33] :: [licz_czyt=2, licz_pis=0]`

5) Wątki pisarzy wypisują komunikat przed wejściem i po wyjściu do/z czytelni. Komunikat powinien mieć postać (początkowe symbole oznaczają: **„==>"** – przed, **„<=="** – po):

   ==> [id wątku, nr powt.] :: [licz_czyt=2, licz_pis=0]

   `==> [P-2, 33] :: [licz_czyt=2, licz_pis=0]`

6) W celu synchronizacji procesów wykorzystać obiekty klasy `Semaphore`.
7) Przerwać wykonywanie wątków potomnych po zadanym czasie np. 10s. (wykorzystać metodę `interrupt()` i `join()`).
8) Działanie programu zademonstrować dla m=5, n=2, K=3.

Zadanie 4.

1) Napisać program współbieżny symulujący działanie ucztujących filozofów.
2) Działanie wątku filozofa polega na wielokrotnym wykonywaniu po sobie ciągów instrukcji odpowiadających tzw. *sprawom własnym* oraz synchronizowanemu spożywaniu posiłków. Sprawy własne polegać maja na wstrzymaniu wątku przez losowy czas w przedziale <5, 15> milisekund.
3) Synchronizowane spożywanie posiłków z wykorzystaniem widelców objawia się wstrzymaniem wątku przez losowy czas w przedziale <1, 5> milisekund oraz wypisaniem stosownych komunikatów podczas rozpoczęcia (>>>) i zakończenia (<<<) spożywania.
4) Komunikat powinien mieć postać:

   >>> [id wątku, nr powt.] :: [fil_przy_stole=2 , w0=1, w1=1, w2=1, w3=0, w4=0]

   `>>> [F-1, 33] :: [fil_przy_stole=2, w0=1, w1=1, w2=1, w3=0, w4=0]`

5) W celu synchronizacji procesów wykorzystać obiekty klasy `Semaphore`.
6) Przerwać wykonywanie wątków potomnych po zadanym czasie np. 10s. (wykorzystać metodę `interrupt()` i `join()`).