1. Proszę przedstawić implementację semafora uogólnionego na poziomie systemu operacyjnego, z preferencją dla:
   (a) przepustowości,
   (b) sprawiedliwości.
2. Poniżej przedstawiony został program dwóch wątków procesu. Proszę wykazać, że w współbieżnym przetwarzaniu tych wątków może dość do zakleszczenia.

```
pthread_mutex_t m1 = PTHREAD_MUTEX_INITIALIZER;
pthread_mutex_t m2 = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t c1  = PTHREAD_COND_INITILIZER;
pthread_cond_t c2  = PTHREAD_COND_INITILIZER;
```

`pthread_mutex_lock(&m1);`	`pthread_mutex_lock(&m2);`
`pthread_cond_signal(&c1);`	`pthread_cond_wait(&c1, &m2);`
`pthread_mutex_unlock(&m1);`	`pthread_mutex_unlock(&m2);`
`pthread_mutex_lock(&m2);`	`pthread_mutex_lock(&m1);`
`pthread_cond_wait(&c2, &m2);`	`pthread_cond_signal(&c2);`
`pthread_mutex_unlock(&m2);`	`pthread_mutex_unlock(&m1);`

3. Proszę wykazać, że użycie zamków w synchronizacji wątków z użyciem zmiennych warunkowych (schemat 1 lub 2) jest warunkiem koniecznym poprawności realizacji przetwarzania.

4. Proszę przedstawić implementację semafora
   (a) binarnego,
   (b) ogólnego,
   (c) uogólnionego,
   (d) dwustronnie ograniczonego,
   za pomocą zamków i zmiennych warunkowych.

5. Proszę przedstawić rozwiązanie problemu
   (a) czytelników i pisarzy,
   (b) pięciu filozofów,
   (c) śpiących fryzjerów,
   za pomocą zamków i zmiennych warunkowych.

6. Proszę przedstawić rozwiązanie problemu czytelników i pisarzy preferujące pisarzy w dostępie do czytelni.

7. Proszę wykazać, że w przedstawionym poniżej rozwiązaniu problemu pięciu filozofów może dojść do zakleszczenia.

```
shared widelec: array[0..4] of Binary_Semaphore := true;

while ... do begin
   myślenie;
   P(widelec[i]);
   P(widelec[(i+1) mod 5]);
   jedzenie;
   V(widelec[i]);
   V(widelec[(i+1) mod 5]);
end;
```

8. Proszę wykazać, że w poniższym rozwiązaniu problemu producenta i konsumenta może dojść do zakleszczenia.

```
const n: Integer := rozmiar bufora;
shared buf: array [0..n-1] of ElemT;
shared wolne:  Semaphore := n;
shared zajęte: Semaphore := 0;
shared mutex:  Binary_Semaphore := 1;
```

Producent

```
local i: Integer =: 0;
local elem: ElemT;
while ... do begin
   produkuj(elem);
   P(mutex);
   P(wolne);
   buf[i] =: elem;
   i =: (i+1) mod n;
   V(zajęte);
   V(mutex);
end;
```

Konsument

```
local i: Integer =: 0;
local elem: ElemT;
while ... do begin
   P(mutex);
   P(zajęte);
   elem =: buf[i];
   i =: (i+1) mod n;
   V(wolne);
   V(mutex);
   konsumuj(elem);
end;
```

9. Proszę przedstawić rozwiązanie problemu ograniczonego buforowania za pomocą semaforów, uwzględniając możliwość współbieżnego działania kilku producentów i kilku konsumentów.

10. Proszę przedstawić definicję monitora do synchronizacji czytelników i pisarzy wg. schematu przedstawionego w module wykładowym.

11. Proszę wykazać niepoprawność rozwiązania problemu śpiących fryzjerów, zaprezentowaną w module wykładowym, jeśli program klienta wygląda następująco:

```
while ... do begin
   P(mutex);
   if l_czek < p then begin
      l_czek := l_czek + 1;
      V(klient);
      P(fryzjer);
      V(mutex);
      strzyżenie;
   end
   else V(mutex);
end;
```

12. Proszę wykazać niepoprawność rozwiązania problemu śpiących fryzjerów, zaprezentowaną w module wykładowym, jeśli program klienta wygląda następująco:

```
while ... do begin
   P(klient);
   P(fotel);
   P(mutex);
   l_czek := l_czek - 1;
   V(fryzjer);
   V(mutex);
   strzyżenie;
   V(fotel);
end;
```