Vytvořte program, který po spuštění vytvoří určitý počet vláken a pomocí operací serializátoru se budou tato vlákna vzájemně synchronizovat.

Jednoduchý serializátor implementujte přímo v programu jako tři funkce:

1. `int getticket(void){ ... }`:
   Výstupní hodnotou této funkce je unikátní číslo lístku, který určuje pořadí vstupu do kritické sekce. První získaný lístek má hodnotu 0, další 1, 2, atd.
2. `void await(int aenter){ ... }`:
   Vstup do kritické sekce, kde parametr aenter je číslo přiděleného lístku funkcí getticket(). Na počátku programu je vstup umožněn jen vláknu s lístkem 0. V kritické sekci může být v daném okamžiku maximálně jedno vlákno.
3. `void advance(void){ ... }`:
   Výstup z kritické sekce, což umožní vstup jinému vláknu přes funkci await() s lístkem o jedničku vyšším, než mělo vlákno kritickou sekci právě opouštějící.

Program bude na příkazovém řádku jako první parametr očekávat počet vláken N, která se mají vytvořit, a jako druhý parametr M celkový počet průchodů kritickou sekcí. Např. program 1024 100 vytvoří 1024 vláken schopných se vzájemně synchronizovat a v kritické sekci se ocitne postupně 100 vláken (některá mohou vícekrát, některá vůbec). Pokud nebude počet vláken nebo počet průchodů zadán, vypíše se způsob použití programu.

V programu v kritické sekci vypisujte jen identifikaci vlákna a číslo lístku, díky kterému vlákno do kritické sekce vstoupilo. Identifikace id by měla být číslo vlákna v intervalu <1, N>. Rozhodně nevypisujte přímo výstupní hodnotu funkce pthread_self() typu pthread_t, jelikož nikde není řečeno, jaký konkrétní typ to má být a jakých hodnot má nabývat. Základní kostra kódu může vypadat následovně:

```
while ((ticket = getticket()) < M) { /* Přidělení lístku */
   /* Náhodné čekání v intervalu <0,0 s, 0,5 s> */
   await(ticket);              /* Vstup do KS */
   printf("%d\t(%d)\n", ticket, id); /* fflush(stdout); */
   advance();              /* Výstup z KS */
   /* Náhodné čekání v intervalu <0,0 s, 0,5 s> */
}
```

Před použitím každé funkce si nejdříve v manuálové stránce pozorně zkontrolujte, zda je její použití bezpečné i v programu s vlákny! Například funkce rand() ve vláknových programech není bezpečná nikdy díky existenci globální proměnné a u funkce usleep() je díky způsobu implementace v Solarisu uvedeno MT-Level: Unsafe, a proto ji nemůžete pro čekání použít.

Místo usleep() lze použít funkci nanosleep() a pro generování náhodných hodnot lze použít rand_r(). Inicializaci parametru seed pro každé vlákno proveďte například na základě aktuálního

času v mikrosekundách (nebo v tomto konkrétním projektu alespoň podle čísla procesu) a jednoznačné identifikace vlákna.

Je zakázáno používat aktivní čekání - tedy testování ve smyčce na hodnotu nějakého příznaku - bude považováno za naprosto špatné řešení!

Hlavní stránka o vláknech v Solarisu je dostupná napsáním příkazu man pthreads, na FreeBSD je to man pthread. Hned na začátku je vidět (mezery za znaménkem - v SYNOPSIS být nemají), jak správně překládat vláknové programy v Solarisu. Při překladu pomocí gcc se parametr -mt nepoužívá.