**Esame di sistemi operativi del 23/11/2012**

**Primo esercizio**

Un ufficio postale ha 4 postazioni, di cui tre dedicate al pagamento dei bollettini postale e uno per la spedizione delle lettere.

Gli utenti, scelta la propria tipologia in maniera random(0 se devono pagare i bollettini 1 se devono spedire una lettera o un pacco) prendono il proprio turno (da 1 a M con M dato in input) e si accomodano in uno delle N sedie (con N dato in input) disponibili.

Quando arriva il proprio turno va al primo sportello libero e lo occupa per un tempo random.

Se non vi sono più sedie disponibili l’utente ritornerà in seguito, la simulazione termina quando il turno raggiunge il valore massimo.

Alla fine della simulazione stampare quanti utenti sono stati serviti da ogni sportello.

Si considerano gli utenti come thread, e per la sincronizzazione utilizzare esclusivamente mutex o semafori.

La vita degli utenti è questa:

* Scelgo la tipologia di utente random(0,1)
* Prendo il turno
* Cerco una sedia, se non ce ne sono libere aspetto sino a che se ne liberi una
* Aspetto finché non si liberi uno sportello
* Pago o spedisco per un tempo random (da 100 a 1000ms)
* Vado via

**Secondo esercizio**

Scrivere un programma in C multi-processo in ambiente linux in cui si simuli l’esercizio precedente considerando questa volta gli sportelli come processi distinti. Questa volta gli sportelli sono abilitati sia alla spedizione che al pagamento, e si conosce il numero di utenti. Quindi:

* Il processo principale dato in input N numero utenti
  + Genera 4 processi, uno per sportello
  + Per ogni figlio Sportello finché il turno <=N

1. Comunica di volta in volta ai vari figli il turno
2. Aspetta che il figlio i-esimo gli comunichi che ha finito
3. Aggiorna turno che andrà da 1 a N

* Il processo i-esimo
  + Chiede al processo principale il turno
  + Finché il turno ricevuto <=N

1. Simula il pagamento o la spedizione per un tempo random da 1 a 5 sec
2. Comunica al processo principale che ha finito
3. Richiede il turno al processo principale