Nome_____ Cognome _____N. di matricola (10 cifre) _____ Riga ____ Col ____

UNIVERSITA' DEGLI STUDI DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
PROVA SCRITTA DI SISTEMI OPERATIVI
ANNO ACCADEMICO 2018/2019
13 settembre 2019

Esercizio -1: Essere iscritti su AlmaEsami per svolgere questa prova.

Esercizio 0: Scrivere correttamente nome, cognome, matricola e posizione in tutti i fogli prima di svolgere ogni altro esercizio. Scrivere esclusivamente a penna senza abrasioni. E' vietato l'uso delle penne cancellabili, della matita, dei coprenti bianchi per la correzione (bianchetto) e la scrittura in colore rosso (riservato alla correzione).
Il compito e' formato da tre fogli, sei facciate compresa questa. Le soluzioni che si vogliono sottoporre per la correzione devono essere scritte negli spazi bianchi di questi fogli. Non verranno corretti altri supporti.
E' obbligatorio consegnare il compito, e' possibile chiedere che esso non venga valutato scrivendo "NON VALUTARE" in modo ben visibile nella prima facciata.
Per svolgere questo compito occorre solo una penna e un documento di identità valido. La consultazione o anche solo la disponibilità di altro materiale comporterà l'annullamento del compito (verrà automaticamente valutato gravemente insufficiente).

Esercizio c.1: Scopo di questo esercizio è di scrivere un monitor `mbuf` che realizzi un buffer limitato dove ogni elemento venga inserito con una molteplicità. In particolare `mbuf` deve fornire le seguenti procedure entry:
`void add(type data, int n);`
che inserisce il dato `data` con molteplicità `n`.
`type get(void);`
che restituisce il primo dato disponibile.
Il monitor deve soddisfare i seguenti vincoli:

- il buffer deve contenere al più `MAXELEM` elementi (ogni operazione add aggiunge un elemento indipendentemente dalla molteplicità `n`)
- i dati vengono consegnati in ordine FIFO
- un dato inserito con molteplicità `n` deve essere ricevuto da `n` processi: il monitor deve attendere che `n` processi chiamino (o abbiano chiamato) la `get` e a quel punto restituisce il dato a tutti e toglie l'elemento dal buffer.

Esercizio c.2:

```
process bohm[i, i=0,1] {
  for (;;) {
    m.pre(i);
    print(i);
    m.post(i);
  }
}
monitor m:
  condition ok[2];
  state = 0;
  procedure entry pre(int n) {
    if (state != n)
       ok[n].wait();
  }
  procedure entry post(int n) {
    state = 1 - state;
    ok[state].signal()
  }
}
```

```
.........

process bohs[i, i=0,1] {
  for (;;) {
    pre(i);
    print(i);
    post(i);
  }
}

void pre(int n) {
........
}

void post(int n) {
........
}
```

Scopo di questo esercizio è di studiare il comportamento del programma composto dai processi bohm e dal monitor m nella colonna di sinistra e di completare il programma sulla destra in modo che abbia lo stesso comportamento ma usi i semafori al posto del monitor e che sia minimale in termini di numero di istruzioni e di variabili utilizzate.

Esercizio g.1: Sia dato in programma che elabori per x millisecondi, compia una operazione di I/O su uno specifico device per y millisecondi quindi faccia una elaborazione finale per x millisecondi e termini.
Si eseguano tre istanze dello stesso programma su un sistema monoprocessore. I tre processi iniziano l'esecuzione al tempo zero e accedono allo stesso device in modo mutualmente esclusivo: le richieste di I/O vengono gestite in modalità FIFO.
Si confronti ora il comportamento di uno scheduler del processore round-robin con valore di time slice m e quello di uno scheduler non preemptive a priorità statica (i tre processi hanno priorità diversa).
A quali condizioni e per quali valori di x, y e m i due scheduler si comportano nello stesso modo?

Esercizio g.2: rispondere alle seguenti domande:
a) perché l'invenzione degli interrupt ha reso i sistemi operativi più efficienti?
b) ha senso utilizzare RAID 5 con due dischi?
c) in quali casi entra in funzione il paginatore in un sistema di memoria virtuale e quando viene richiamato l'algoritmo di rimpiazzamento?
d) quali sono i vantaggi e quali gli svantaggi dell'utilizzo di librerie dinamiche?