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
18 giugno 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: passaggio di dati *uno o centomila ma non nessuno*.
Scrivere un monitor con due procedure entry:
```
put(T dato);
T get(void);
```
Esistono due tipi di processo: gli scrittori che chiamano *put* e i lettori che chiamano la *get*.
La *put* deve fare in modo che tutti i lettori in attesa, qualora ve ne siano, ricevano il dato passato come parametro. Se non ci sono lettori in attesa la *put* deve attendere la prima chiamata della procedure entry *get* e trasferire il dato. Se ci sono già scrittori in attesa, il nuovo scrittore si mette in coda (ordine FIFO).
La *get* quindi se c'è uno scrittore in attesa deve ritornare il dato del primo scrittore in attesa (e sbloccare lo scrittore), altrimenti deve attendere la prossima chiamata della procedure entry *put*.
Domanda ulteriore: è possibile implementare il monitor con una sola variabile di condizione? Perché?

Esercizio c.2:

```
class semaphore {
  int value;
  int blocked = 0;
  binary_semaphore mutex(1);
  binary_semaphore sem(0);

void init(int initval) {
  value = initval;
}
```

```
void P() {
  mutex.P();
  if (value == 0) {
    blocked++;
    mutex.V();
    sem.P();
    blocked--;
  }
  value--;
  mutex.V();
}
```

```
void V() {
  mutex.P();
  value++;
  if (blocked > 0)
    sem.V();
  else
    mutex.V();
}

} // class semaphore
```

In questa implementazione di semafori generali dati semafori ordinari:
a) viene usata la tecnica di passaggio del testimone (*passing le batôn*), spiegare come è usata e quale effetto ha.
b) questa implementazione non garantisce la FIFONESS di riattivazione dei processi bloccati, mostrare perché.
c) modificare il codice, mantenendone la struttura, per ottenere una soluzione che risolva il problema del punto 2.

Esercizio g.1: Costruire Grafi di di Holt che abbiano tutte le seguenti proprietà:

- la situazione sia di Deadlock
- non sia sufficente eliminare un processo perché il Deadlock venga risolto
- ogni processo abbia esattamente una richiesta in sospeso
- il grafo sia connesso (ci sono archi fra nodi di partizioni diverse comunque si partizioni l'insieme dei nodi)

Caso A: costruire (se possibile) un grafo con 5 processi e 5 classi di risorse.
Caso B: costruire (se possibile) un grafo con 5 processi e 4 classi di risorse.
Caso A: costruire (se possibile) un grafo con 4 processi e 5 classi di risorse.

Esercizio g.2: rispondere alle seguenti domande:
a) Dati 4 dischi da 1GiB, si possono memorizzare più dati usando una organizzazione RAID10 o RAID5? Perché?
b) Dimostrare che se un algoritmo di rimpiazzamento è a stack allora non può soffrire della anomalia di Belady
c) Quali sono le differenze fra un device driver di una unità con funzionalità di DMA e quello di una unità senza DMA?
d) Come vengono memorizzati i link simbolici in un filesystem?