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 2017/2018
28 maggio 2018

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: La struttura di sincronizzazione *riempisvuota* ha le seguenti caratteristiche:
-) entrano MAX processi, e fino a che non è piena nessun processo può uscire
-) i processi devono uscire in ordine inverso rispetto all'ordine di arrivo.
Quando un processo vuole usare la struttura *riempisvuota* chiama la seguente funzione:

```
def riempisvuotasynch():
   riempisvuota.entra(getpid())
   riempisvuola.esci(getpid())
```

Scrivere il monitor *riempisvuota*.

Esercizio c.2: In un sistema ci sono 26 processi server proc['a'], proc['b'] .... proc['z'] che usano message passing asincrono. Ogni processo è l'unico che può stampare la propria lettera ed esegue il seguente codice:

```
proc[x]: x='a',...,'z'
   while True:
      (c, string) = arecv(*)
      print(x)
      if (len(string) > 1)
         asend(proc[string[0]], (x, string[1:]))
```

e i clienti che vogliono stampare una stringa (non vuota) usano la funzione:

```
def procstampa(s):
   asend(proc[s[0]], (None, s))
```

Se un cliente alla volta chiama la procstampa il sistema produce il risultato voluto ma se malauguratamente un cliente chiama la procstampa mentre è in corso un'altra stampa l'output delle due operazioni può mischiarsi (se il primo cliente ha chiesto di stampare "ciao" e il secondo "mondo" l'output potrebbe essere "cmionadoo").

Senza aggiungere ulteriori processi e senza modificare il codice eseguito dai clienti correggere il codice dei server in modo che il programma concorrente funzioni correttamente in ogni caso (nell'esempio precedente venga stampato prima "ciao" e poi "mondo", o viceversa, ma ogni stringa venga interamente stampata prima che inizi la stampa della successiva).

Esercizio g.1: Ci siano in un sistema due processi P1 e P2 identici che iniziano contemporaneamente. Entrambi i processi computano per 4 ms, fanno I/O per 2 ms e poi elaborano nuovamente per 4 ms. L'I/O dei due processi avviene sulla stessa unita' con accesso FIFO. Il sistema usa uno scheduler di tipo round robin. Calcolare lo schedule per ogni scelta della lunghezza del quanto di tempo e quale sia la scelta (o siano le scelte) migliori. E quale è il limite massimo di durata dello schedule (considerando il time slice come numero reale positivo)?

Esercizio g.2: Rispondere alle domande seguenti:
a) A cosa serve e quando viene eseguito un algoritmo di rimpiazzamento?
b) Come si calcola la lunghezza massima di un file che si può memorizzare su un file system di tipo ext2?
c) A cosa serve il meccanismo del sale (salt) nella memorizzazione delle password criptate? E se il salt funziona, perché le password criptate in Linux vengono memorizzate nel file "/etc/shadow" e non in "/etc/passwd"?
d) Cosa occorre fare per evitare il deadlock quando lo stato calcolato dall'algoritmo del banchiere risulta unsafe?