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

UNIVERSITA' DEGLI STUDI DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
CORSO DI SISTEMI OPERATIVI - ANNO ACCADEMICO 2009/2010
PROVA DI CONCORRENZA – 09 dicembre 2009

Esercizio -1: essersi iscritti correttamente 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 due fogli, quattro facciate compresa questa. Le soluzioni che si vuole 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.

Esercizio 1: siano dati tre processi “stampa” come segue:

```
stampa(x):(x=A,T,R) process
while (true) {
        taratata.synchro(x);
        print(x);
}
```

scrivere il monitor taratata con una sola procedure entry synchro() che consenta di avere in output una sequenza infinita di TARATATA (i.e. TARATATATARATATATARATATATARATATATARATATA.....)

Esercizio 2: Uno scheduler cooperativo prevede che ogni processo chiami la funzione yield() quando puo' cedere il controllo agli altri processi.

La funzione yield deve quindi attivare un altro processo e bloccare il chiamante.

Facendo uso di semafori scrivere tre funzioni init, yield, fini tutte prive di parametri. Init viene chiamata da un processo che vuole iniziare la propria esecuzione, yield e' quella descritta sopra e fini viene chiamata da un processo subito prima di terminare.

Un solo processo alla volta deve essere in esecuzione (fra quelli che hanno chiamato init e non ancora fini), yield deve fare avvicendare I processi in ordine ciclico (come se fossero in una coda circolare, il processo che fa yield deve essere riattivato dopo che tutti quelli in attesa hanno avuto modo di essere eseguiti e chiamare nuovamente yield o fini).

Esercizio 3:

Un servizio di message passing “smemorato” talvolta dimentica il destinatario del messaggio. In tale caso il messaggio viene consegnato al mittente invece che al destinatario.

Questo comportamento puo' essere rappresentato come segue. Sia dato un servizio di message passing asincrono: asend/arecv, random() una funzione di generazione di numeri causali nell'intervallo [0,1[e CONST una constante reale << 1.

```
smemsend(msg, dest)
   if (random()<CONST):
      asend(msg, getpid())
   else:
      asend(msg, dest)
smemrecv(sender)
      arecv(sender)
```

Implementare il servizio si message passing asincrono (asend/arecv) utilizzando il servizio di message passing “smemorato”