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 2016/2017
29 maggio 2017

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: Un professore di Sistemi Operativi organizza una partita a rubabandiera ai giardini margherita come evento di fine corso. Partecipano alla gara due squadre da MAX studenti ognuna (gli studenti sono numerati da 0 a MAX-1, le squadre si chiamano A e B). Il professore puo' chiamare 1 studente (rubabandiera classico), 2 studenti (uno sulle spalle dell'altro), 3 studenti (seggiolino) o quattro studenti (aeroplano).
Il processo professore e' il seguente:

```
A=0
B=1
process prof:
  rb.nuovapartita()
  while True:
    punteggio = rb.chiama(listarandom(MAX)) # genera una lista di 1, 2, 3 o 4 elementi
    print(punteggio[A], ":", punteggio[B])
    if max(punteggio) == 10:
      break
```

I processi studente/studentessa sono rappresentati dal seguente codice

```
process studente(squadra, numero):
  while True:
    if rb.pronto(squadra, numero):
      break
    rb.allabandiera(squara, numero)
```

Regole (del modello): il prof inizia la partita con *nuovapartita* ponendo il risultato iniziale sullo 0 a 0. Durante la partita pensa la lista dei numeri ma sblocca gli studenti solo se tutti sono in *pronto*, altrimenti aspetta.
Vince un punto la squadra che per prima arriva alla bandiera (tutti i componenti chiamati dal prof sono *allabandiera*).
La procedure entry *chiama* restituisce il punteggio attuale. La procedure entry *pronto* restituisce 0 normalmente, un valore diverso da zero indica il termine della partita.

Esercizio c.2: sia dato un servizio di message passing asincrono distratto. Questo servizio si comporta come un servizio di message passing asincrono ma talvolta dimentica la destinazione. E' però possibile indicare un processo come “ufficio messaggi smarriti” al quale verranno recapitati tutti i messaggi per i quali il servizio distratto ha dimenticato la destinazione.

```
Void dmsgsend(pid_t dest, msg_t msg);    //si comporta come amsgsend ma può dimenticare la desticazione
msg_t dmsgrecv()                    //si comporta come amsgrecv(*)
void dset_lost_n_found(pid_t pid);     //indica il processo per i messaggi smarriti.
```

Usando il servizio “distratto” e un processo “ufficio messaggi smarriti”, implementare un servizio di message passing standard (senza la selezione del mittente in ricezione, la amsgrecv riceve da qualsiasi mittente).

Esercizio g.1: Trovare una stringa di riferimenti infinita (che usi un numero finito di pagine), se esiste, per la quale l'algoritmo LIFO (per il quale la pagina vittima è l'ultima caricata) e MIN si comportino esattamente nello stesso modo.
Trovare una stringa di riferimenti infinita (che usi un numero finito di pagine), se esiste, per la quale l'algoritmo LIFO (per il quale la pagina vittima è l'ultima caricata) e LRU si comportino esattamente nello stesso modo.

Esercizio g.2: Rispondere alle domande seguenti:
a) Un i-node di un file system tipo ext2 per un errore viene riportato un numero di link maggiore del reale. Cosa può succedere se si continua ad usare il file system? (perché?) E se il numero di link errato fosse al contrario inferiore al reale cosa potrebbe succedere? (perché?)
b) Perché è necessario usare spinlock in sistemi multiprocessore per implementare kernel di tipo simmetrico (SMP)?
c) Perché nei sistemi reali l'algoritmo di rimpiazzamento second chance (orologio) viene preferito a LRU sebbene il primo non sia a stack e il secondo sì?
d) Perché revocare un'autorizzazione espressa come capability è più difficile che revocare lo stesso diritto quando espresso come access control list?