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
15 gennaio 2020

**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**: Il monitor "semaforo con timeout" semtimeout deve fornire tre procedure entry:

```
void V(void)
boolean P(unsigned int timeout)
void tick(void)
```

Vale l'invariante dei semafori generali. La funzione tick viene richiamata ad ogni millisecondo. Un processo messo in attesa a causa di una P attende al massimo il numero di millisecondi indicato nel parametro.
Se un processo viene riattivato per timeout la P ritorna valore vero, altrimenti falso. Le operazioni V devono riattivare i processi in attesa in ordine FIFO.
Scrivere semtimeout.

**Esercizio c.2:** Dato un servizio di message passing asincrono scrivere un servizio di message passing sincrono a spedizione multipla (senza fare uso di processi server). Devono essere previste due funzioni:

```
mulsend(pid_t destination, T msg, int times)
T mulrecv(pid_t sender)
```

La chiamata mulsend spedisce il messaggio `msg` al processo `destination` `times` volte. Solo quando il messaggio è stato ricevuto `times` volte da `destination` l'esecuzione della funzione `mulsend` viene completata.
Il ricevente può indicare il valore zero come `sender` nella `mulrecv` per indicare che vuole ricevere un messaggio da qualsiasi mittente.

**Esercizio g.1:** Si consideri l'algoritmo del banchiere a tre valute. Si prenda in considerazione una situazione nella qualetre processi p1, p2, p3 abbiano un massimo credito disponibile rispettivamente di (3, 2, 3), (3, 3, 2), (2, 3, 3), cioé peresempio p1 può prendere in prestito 3 unità della prima e della terza valuta ma solo 2 della seconda. In un certo istante p1 ha 2 unità della prima valuta, p2 ha 2 unità della seconda, p3 ha 3 unità della terza e nessuna altra risorsa è stata assegnata. Qual è il capitale iniziale minimo che consente allo stato di essere safe? (considerare tutti i casi possibili e spiegare bene il procedimento.

**Esercizio g.2:** rispondere alle seguenti domande:
a) perché uno scheduler round-robin può essere inadatto a gestire processi con I/O multimediale?
b) per risolvere un problema di trashing è necessario terminare forzatamente dei processi o è sufficiente bloccare l'esecuzione di qualche processo e riattivarli successivamente? Perché?
c) Con lo stesso numero di dischi, RAID1 o RAID5 consente di memorizzare più dati? Perché? Quale è più sicuro? Perché?
d) Perché è difficile revolacre una autorizzazione fornita tramite una *capability*?