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
15 gennaio 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: Un semaforo pesato è una struttura di sincronizzazione con due primitive P(w) e V(w) (oltre al costruttore di inizializzazione). I parametri delle operazioni P e V sono i pesi dell'operazione. L'invariante dei semafori pesati richiede che la somma dei pesi delle operazioni P completate sia sempre inferiore o uguale alla somma dei pesi delle operazioni V completate più il valore di inizializzazione. Scrivere il monitor `wsem` in grado di implementare un semaforo pesato fair: i processi che si bloccano a causa di una P che invaliderebbe l'invariante devono essere riattivati nell'ordine in cui sono stati sospesi (FIFO).

Esercizio c.2: Dato un servizio di message passing asincrono e senza fare uso di processi server:
a) implementare un servizio di message passing sincrono a ricezione multipla. Questo servizio prevede due funzioni con la seguente interfaccia:

```
ms_send(pid dest, msg_t msg)
ms_recv(int n, pid *senders, msg_t *msgs)
```

L'operazione di ricezione deve attendere n messaggi, provenienti dai mittenti indicati nel vettore `senders` (ogni elemento può essere ANY/*) e metterli ordinatamente nel vettore `msgs` (entrambi i vettori saranno stati opportunamente allocati e dimensionati dal chiamante). I processi mittenti degli n messaggi devono rimanere in attesa fino a che la `ms_recv` non può essere completata.
b) analizzare i casi di deadlock che possono accadere in base alla definizione del servizio di message passing sincrono a ricezione multipla.

Esercizio g.1: Si consideri l'algoritmo del banchiere a tre valute. Si prenda in considerazione una situazione nella quale tre processi p1, p2, p3 abbiano un massimo credito disponibile rispettivamente di (3, 2, 3), (3, 3, 2), (2, 3, 3), cioé per esempio 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 valute, 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).

Esercizio g.2: Rispondere alle domande seguenti:
a) Ci sono spinlock (strutture di sincronizzazione simili al Test@Set) nel kernel Linux? Perché?
b) In un file system tipico di UNIX (ext2/3, bffs, etc) l'accesso diretto è meno efficiente quando il file è di grandi dimensioni. E' vero o falso? Perché?
c) Per risolvere una situazione di trashing di memoria occorre terminare forzatamente un processo. E' vero o falso? Perché?
d) Dato un sistema monoprocessore che elabora dati in modo batch, cosa cambia se si usa uno scheduler round-robin al posto di uno FIFO?