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 2021/2022
18 gennaio 2023

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: Scrivere il monitor `synmsg` che consenta uno scambio di messaggi fra due processi in maniera sincrona.
Il processo produttore esegue il seguente codice.

```
producer: process:
    while True:
        msg_t msg = produce_new_msg()
        synmsg.send(&msg)
```

e il processo consumatore:

```
producer: process:
    while True:
        msg_t msg
        synmsg.recv(&msg)
        consume_msg(&msg)
```

Come si vede le procedure entry hanno come parametro l'indirizzo del messaggio:

```
procedure entry void send(msg_t *msg)
procedure entry void recv(msg_t *msg)
```

Il monitor deve trasferire il contenuto del messaggio direttamente fra i processi usando la funzione:

```
void copymsg(msg_t *src, msg_t *dest)
```

(Il monitor non deve avere buffer di tipo msg_t ma solo variabili di tipo puntatore a msg_t.)

Esercizio c.2: Facendo uso di semafori scrivere un funzione wait4 che faccia proseguire i processi a blocchi di quattro: il primo processo che chiama la wait4 si deve fermare, così come il secondo e il terzo. Il quarto processo deve far proseguire tutti e quattro i processi. In uguale modo l'ottavo processo che chiama wait4 risveglierà anche il quinto, il sesto e il settimo.
SI chiede:
* che l'implementazione non faccia uso di code o di altre strutture dati ma solamente di contatori (e ovviamente semafori)
* che la soluzione faccia uso del passaggio del testimone per garantire che vengano riattivati i processi corretti e non altri.

Esercizio g.1: Siano dati due processi in esecuzione in un sistema monoprocessore e gestiti da uno scheduler round-robin.
I due processi sono gli unici nel sistema e usano la stessa unità di I/O gestita in modo FIFO.

```
PA: 1ms CPU, 2ms I/O, 1ms CPU, 8ms I/O, 1ms CPU
PB: 2ms CPU, 1ms I/O, 8ms CPU, 1ms I/O, 1ms CPU
```

Quale è il tempo minimo e quale il tempo massimo impiegato dal sistema per completare l’elaborazione dei due processi al variare della lunghezza del quanto di tempo e della posizione iniziale dei processi nella ready queue (PA precede PB o viceversa).

Esercizio g.2: rispondere alle seguenti domande (*motivando opportunamente le risposte!*):
a) Quali sono gli interrupt hardware più frequenti e quali le trap (interrupt software) più frequenti?
b) A cosa serve il Translation Lookahead Buffer?
c) Quali problemi può porre l'implementazione di un Sistema Operativo con il supporto di memoria virtuale e di controller di I/O di tipo DMA?
d) Qual è la complessità computazionale dell'algoritmo del banchiere monovaluta e quella del banchiere multivaluta?