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
17 luglio 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: In una conferenza il coordinatore decide l’ordine degli oratori e li chiama uno ad uno per poter fare il proprio intervento. Se l’oratore arriva in ritardo (non sta attendendo al momento della chiamata) perde il diritto di poter parlare.

```
Coordinatore: process
    while True:
        chiamato = next(); //next è la funzione che restituisce il nome del prossimo oratore
        print(“Chiamo ora a parlare “,chiamato);
        if (conf.chiama(chiamato))
            print(“ringrazio “, chiamato,” per la relazione”);
        else
            print(“mi dispiace che “, chiamato, “non sia presente”);

Oratore[nome]: for nome in set_of_speakers
    if conf.arrivato(nome):
        //presentazione
        conf.finepresentazione(nome)
```

Scrivere il monitor conf. La funzione chiama aspetta che il relatore chiamato abbia completato l’intervento, se prensente, e restituisce vero altrimenti restituisce falso. La funzione arrivato segnala la presenza e pone il relatore in attesa del proprio turno. Se il relatore è già stato chiamato ed era assente restituisce falso.

Esercizio c.2: Sia dato un servizio di message passing a diffusione che fornisce due funzioni:

```
void bsend(msg_type msg)
msg_type brecv(void)
```

I messaggi spediti con la bsend vengono ricevuti da tutti i processi e la brecv riceve i messaggi spediti da ogni mittente.
I messaggi di ogni mittente vengono ricevuti il ordine FIFO.
E’ possibile con il servizio a diffusione implementare un servizio di message passing asincrono? Se sì fornire l’implementazione, se no (di)mostrare l’impossibilità.

Esercizio g.1: Sia data la seguente funzione:

```
    void shift_first(char *a) {
        int i;
        for (i=0; i<95; i++)
            a[i * 1024] = a[(i+5) * 1024];
    }
```

In un sistema con pagine da 1K ci sono tre processi che stanno contemporamente eseguendo la funzione shift_first. I tre processi non condividono dati. Nell’ipotesi che sia il codice sia lo stack siano in segmenti non gestiti dalla memoria virtuale calcolare il working set complessivo con ampiezza di finestra 10, 20 e 30. Se ci sono 45 frame di memoria siamo in presenza di trashing o no?

Esercizio g.2: Rispondere alle domande seguenti:
a) la ricostruzione della coerenza mediante tecnica di journaling risolve solo alcuni tipi di incoerenze, quali non vengono gestite?
b) applicando la paginazione con la segmentazione aumenta la frammentazione interna (rispetto alla paginazione senza segmentazione). Questa affermazione è vera o falsa, perché?
c) Quale è la differenza fra gli algoritmi di scheduling “Shortest Time First” e “Shortest Remaining Time First”?
d) Perché il meccanismo del “sale” (salt) rende più complesso l’attacco di tipo dizionario alle password?