Nome_____ Cognome _____N. di matricola (10 cifre) _____ Riga ___ Col _____

UNIVERSITA' DEGLI STUDI DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
CORSO DI SISTEMI OPERATIVI - ANNO ACCADEMICO 2008/2009
CONCORRENZA – 13 gennaio 2010

Esercizio -1: essersi iscritti correttamente 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 due fogli, quattro facciate compresa questa. Le soluzioni che si vuole 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.

Esercizio 1: Scrivere un monitor chiamato lifobuf che implementi un buffer limitato (di MAX elementi) che restituisca gli elementi in ordine LIFO. Esistono numerosi processi produttori e numerosi consumatori che utilizzano lilobuf:

```
producer: array[1...NPROD] of processes
  while (1) {
    x=produce();
    lilobuf.put(x);
  }
consumer: array[1...NCONS] of processes
  while (1) {
    x=lilobuf.get();
    consume(x);
  }
```

Se vi e' spazio nel buffer I produttori non vengono sospesi. Quando il buffer e' pieno I produttori devono attendere.

I consumatori devono prendere sempre l'ultimo elemento, o dal buffer o sbloccare l'ultimo processo in attesa.

Esercizio 2: Dato un servizio di message passing asincrono e senza utilizzare processi server realizzare un servizio di message rmp con le seguenti caratteristiche:

rmpsend(dest,msg) spedisce il messaggio al processo desinatario solo se questo e' bloccato in attesa di un messaggio (ha chiamato la rmsrecv e sta aspettando). In caso contrario la chiamata non ha alcun effetto.

rmprecv(sender) riceve un messaggio dal mittente specificato (che non puo' essere '*').

Esercizio 3: Siano date le seguenti funzioni:

i. f(a,b,c)=< binrandom()==0 ? (a=b, b=c) : (c=b,b=a) > /*binrandom genera un numero binario random */
ii. $f_k(x)$ = <return x = k * x> con k costante reale
iii. $f_k(x)$ = <return x = k * x> con k costante complessa

Quali fra queste funzioni, e per quali valori di K ove previsto, possono essere utilizzate per realizzare meccanismi supporto di sezioni critiche come la Test&Set?

Nome_____ Cognome _____N. di matricola (10 cifre) _____ Riga ___ Col _____

UNIVERSITA' DEGLI STUDI DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
CORSO DI SISTEMI OPERATIVI - ANNO ACCADEMICO 2008/2009
PARTE GENERALE – 13 gennaio 2010

Esercizio -1: essersi iscritti correttamente per svolgere questa prova.

Esercizio 0: Scrivere correttamente nome, cognome, matricola e posizione prima di svolgere ogni altro esercizio.

Esercizio 1: Il file system di MINIX usa blocchi di 1K byte che vengono numerati con interi (puntatori) a 16 bit. Ogni i-node ha 9 puntatori: 7 puntatori diretti, uno indiretto e uno indiretto-doppio.

1. Qual e' la lunghezza massima dei file in tale file system?
2. A partire da un i-node, indicare quanti blocchi occorre leggere per accedere a un byte in una posizione random nel file. (dalla posizione ??? alla posizione ???, ??? accessi, dalla posizione yyy...)
3. Data L la lunghezza in byte di un file F, scrivere la funzione block_len(L) che associa a L il numero dei blocchi necessari per memorizzare il file F? (il file F e' completamente allocato, non ha “buchi”).

Esercizio 2: Sia dato un algoritmo di minimizzazione delle seek chiamato Odd-Even Look: l'algoritmo soddisfa tutte le richieste pendenti per I cilindri di numero dispari nella direzione crescente, quando non vi sono piu' richieste dispari raggiunge la richiesta con il massimo numero pari e soddisfa durante il percorso di ritorno (descrescente) le richieste relative ai cilindri di numero pari. Quando non vi sono piu' richieste che soddisfano questa condizione l'algoritmo riparte in senso crescente dalla richiesta con il minimo numero dispari di cilindro. Se arriva una richiesta pari quando la testina sta andando in direzione crescente o viceversa dispari quando la direzione e' descrescente, la richiesta viene memorizzata per essere soddisfatta quando verra' invertita la direzione. Se arriva una richiesta dispari durante la scasione in direzione crescente o pari in direzione decrescente ma per cilindri gia' scanditi, la richiesta viene memorizzata e postposta per il giro successivo.

1. Odd-Even Look e' sempre piu' efficiente di C-LOOK (in termini di tempo totale di seek), opuure e' sempre meno efficiente o infine esistono casi in cui e' piu' efficiente e casi in cui e' meno efficiente?
2. E' uniforme? I.e. esistono cilindri scanditi piu' frequentemente di altri?

Documentare appropriatamente la risposta.

Esercizio 3: (obbligatorio) Dato N e' il numero di matricola del candidato, calcolare M=N%5. Dato il capitolo del corso di Sistemi Operativi contrassegnato dal numero M nella lista che segue, indicarne in modo schematico il contenuto e I concetti principali.

0: Gestione della Memoria Principale
1: Scheduling della CPU
2: Gestione delle Risorse
3: File System
4: Gestione della Memoria Secondaria