Nome/cognome ___________________________ N. di matricola (10 cifre) _______________ Posizione: Riga ___ Col ___

UNIVERSITA' DEGLI STUDI DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
CORSO DI SISTEMI OPERATIVI - ANNO ACCADEMICO 2006/2007
CONCORRENZA - 08 Febbraio 2007

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: Un parcheggio ha una capacita' di P postii e una coda per l'ingresso che al massimo contiene C auto. L'auto che vuole entrare nel parcheggio si comporta in questo modo:

```
auto: process (id e' l'identificativo del processo)
....
okay=parking.accoda(id)
if (okay) {
      parking.entra(id);
      .... rimane parcheggiata ...
      parking.esce(id);
}
```

Se il parcheggio e' pieno non si puo' entrare, le macchine in attesa devono aspettare che una delle auto parcheggiate esca.
Se la coda di ingresso e' piena, accoda restituisce "falso" e quindi la macchina se ne va.
Le macchine devono entrare nel parcheggio nell'ordine con cui si sono accodate (nell'ordine di esecuzione della funzione accoda).
Scrivere il Monitor parking.

Esercizio 2:
Implementare un meccanismo di sezione critica utilizzando la seguente funzione atomica:
F(X,Y) { X >>= Y; Y = 0>
dove >> e' l'usuale operatore di bit shifting del C.
Esempio: 101101 >> 2 ritorna 001011 in quanto in X >> Y gli Y bit meno significativi vengono perduti, tutti gli altri vengono shiftati a destra di Y posizioni e gli Y nuovi bit piu' significativi sono messi a 0

Esercizio 3 (semafori imbricati):
Un semaforo imbricato e' un tipo di dato astratto con le primitive seguenti, dove abbiamo supposto che a ogni processo P sia associato un insieme di semafori “posseduti” da P.

$init_m$(S,n)	inizializza il semaforo S al valore n
P_m(S)	decrementa il valore del semaforo; se il precedente valore era positivo aggiunge S all'insieme dei semafori posseduti da P; altrimenti sospende il processo corrente rilasciando tutti i semafori da esso posseduti (ovvero facendo una V_m su tutti i semafori da esso posseduti)
V_m(S)	incrementa il valore del semaforo; se ci sono processi sospesi risveglia il primo processo P (se esiste) che e' in grado di acquisire atomicamente tutti i semafori precedentemente posseduti al momento della sospensione

Esempio: supponiamo che S1 valga 1, S2 valga 1 e S3 valga 0, che processo P esegua { P_m(S1); P_m(S2); P_m(S3); } e che successivamente un processo Q esegua V_m(S3). Il sistema attraversera' i seguenti stati:

istruzione eseguita	S1	S2	S3	stato di P	stato di Q
	1	1	0	running	
P: P_m(S1)	0	1	0	running	
P: P_m(S2)	0	0	0	running	
P: P_m(S3)	1	1	-1	suspended su S3	
	1	1	-1	suspended su S3	running
Q: V_m(S3)	0	0	0	running	running

a) **[FACILE]** Implementare le primitive dei semafori tradizionali utilizzando semafori imbricati
b) Implementare le primitive dei semafori imbricati utilizzando semafori tradizionali

Nome/cognome ________________________ N. di matricola (10 cifre) __________________ Posizione: Riga __ Col___

UNIVERSITA' DEGLI STUDI DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
CORSO DI SISTEMI OPERATIVI - ANNO ACCADEMICO 2006/2007
PARTE GENERALE - 08 Febbraio 2007

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:

Mostrare l'esecuzione con 4 frame dell'algoritmo di rimpiazzamento descritto qua sotto sulla stringa di riferimenti
1 2 1 3 4 3 1 5 3 1 6 7 8 1 7 7 8 7 8 7 8 8 5 6 5 3 3 2 1 4 6

L'algoritmo funziona nel modo seguente:

Se tutte le pagine in memoria sono state accedute negli ultimi 2*#frames accessi allora la pagina vittima e' la pagina acceduta meno frequentemente negli ultimi 2*#frames accessi. A parita' di accessi scegliere la pagina caricata meno recentemente. Se invece ci sono pagine in memoria che non sono state accedute negli ultimi 2*#frames accessi, allora la pagina vittima e' la pagina scelta dall'algoritmo LRU.

[DIFFICILE] L'algoritmo e' a stack? Se si', dare una dimostrazione. Altrimenti fornire un controesempio.

Esercizio 2:

Si confrontino le esecuzioni degli algoritmi SJN (non preemptive) e SRTF
(preemptive) nell'esecuzione dei seguenti programmi.

```
P1:   for (i=0;i<3;i++) {
          read(1ms);
          compute(4ms);
          write(2ms);
      }

P2:   for (i=0;i<3;i++) {
          read(1ms);
          compute(1ms);
          write(2ms);
      }
```

Le letture e le scritture avvengono su un unico device (FIFO).

I tempi dei context switch e di valutazione del for si intendono tralasciabili ai fini didattici.

Esercizio 3: Sia x la terz'ultima cifra del numero di matricola e y la penultima cifra. Descrivere il metodo (x*10+y)%4 senza dimenticarsi di porre in evidenza limiti, vantaggi e svantaggi rispetto a tutti gli altri metodi elencati.

0) Deadlock avoidance
1) Deadlock detection per mezzo di grafi wait-for
2) Deadlock detection per mezzo di grafi di Holt
3) Deadlock prevention