Nome/cognome _____ N. di matricola (10 cifre) _____ Posizione: Riga _____ Col _____

UNIVERSITA' DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
CORSO DI SISTEMI OPERATIVI - ANNO ACCADEMICO 2004/2005
COMPITO CONCORRENZA – 22 Luglio 2005

Esercizio -1: essersi iscritti correttamente per svolgere questa prova.
Esercizio 0: Su entrambi i fogli, scrivere correttamente nome, cognome, matricola e posizione prima di svolgere ogni altro esercizio.

Esercizio 1
Un Internet Cafe' dotato di 2N postazioni è gestito da un laureato in informatica. Per promuovere l'uso di software libero, N postazioni sono basate su GNU/Linux, mentre le altre sono basate su Windows. La politica di gestione è la seguente. All'arrivo, un cliente invoca la procedure entry

p.e. int cafè_enter(int time, int maxdelay, int type)

dove **time** è il "tempo di macchina" che vuole acquistare, **maxdelay** è il tempo massimo che il cliente è disposto ad attendere se tutte le macchine sono occupate, e **type** assume i valori 1=linux, 2=windows, 3=indifferente.
Se esiste una macchina libera del tipo prescelto, il cliente inizia subito ad utilizzarla. Altrimenti, il sistema calcola il ritardo in base ai tempi dichiarati dai clienti che utilizzano le macchine e i clienti in coda; se questo ritardo è inferiore a **maxdelay**, il cliente viene messo in coda fino a quando una macchina del tipo prescelto si libera; a quel punto, la procedure entry ritorna un identificatore della macchina (ad es., incluso nel range [0..2N-1]. Altrimenti, la procedure entry ritorna un valore negativo.
ìQuando il cliente termina, invoca la :

p.e. void cafè_exit(int index)

dove index è l'identificatore della macchina utilizzata. Scrivere il **monitor** InternetCafè.
Nota: i clienti sono assolutamente precisi nello sfruttare i tempi di macchina, quindi restituiscono la macchina esattamente al tempo previsto. Non è quindi necessario impostare "timer" per risolvere l'esercizio, basta sbloccare clienti in coda durante la **cafe_exit**.

Esercizio 2

```
shared val = 0;
shared Semaphore sp =
  new Semaphore(2);
shared Semaphore sq =
  new Semaphore(1);
shared Semaphore mutex =
  new Semaphore(1);
```

```
process P {
  int kp = 2;
  while (kp > 0) {
    sp.P();
    mutex.P();
    val = val*3;
    sq.V();
    kp--;
    mutex.v();
  }
}
```

```
process Q {
  int kq = 3;
  while (kq > 0) {
    sq.P();
    mutex.P();
    val = val+1;
    sp.V();
    kq--;
    mutex.v();
  }
}
```

a) Al termine di questo programma, quali sono i valori possibili della variabile condivisa **val**?
b) E' possibile che i processi P o Q restino bloccati indefinitamente?

Esercizio 3
Sia dato un servizio di semafori fair con le solite chiamate P e V. Sulla base di esso implementare un servizio di semafori unfair (implementare cioe' due funzioni UP e UV che facciano uso di P e V). L'invariante dei semafori risulta sempre verificata sia nel servizio fair sia in quello unfair, ma la UV quando sblocca un processo lo sceglie casualmente fra quelli in attesa.