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 2006/2007
CONCORRENZA - 15 Giugno 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: A Vallettopoli esistono i valletti e i dirigenti. I dirigenti sono rigorosamente organizzati per ruolo in maniera gerarchica: 1 direttore generale, 3 direttori di rete, 9 direttori di studio. Per poter apparire in televisione ogni valletto deve collezionare tre raccomandazioni da tre dirigenti con ruoli diversi. Per non scontentare nessuno, il direttore generale puo' fornire al piu' nove raccomandazioni, i direttori di rete al piu' 3 e i direttori di studio una sola.

a) Scrivere il monitor Vallettopoli che rispetti la seguente interfaccia:

```
monitor Vallettopoli {
   p.e.  void chiedi_raccomandazione(pid dirigente); // il valletto si sopende fino a quando non ottiene la raccomandazione
   p.e.  int appari(void); // il valletto fa la sua comparsata in televisione, “rilasciando” le raccomandazioni ottenute
                           // ritorna 1 (senza rilasciare le raccomandazioni) nel caso il valletto non sia stato raccomandato
                           // come da protocollo; ritorna 0 altrimenti
}
```

I valletti il seguente codice:

```
process valletto() {
  int dirigente[3];
  scegli_tre_dirigenti(dirigenti);  // inizializza dirigenti[] con tre pid opportuni scelti a caso, uno per ruolo, in ordine casuale
  for (int j=0; j<3; j++) { Vallettopoli.chiedi_raccomandazione(dirigente[j]); }
  return Vallettopoli.appari();
}
```

b) Quanti valletti sono necessari affinche' ci possa essere deadlock?

c) E' possibile cambiare la specifica (non il prototipo) della scegli_tre_dirigenti() in modo tale che il deadlock sia prevenuto?

Esercizio 2: Join Calculus binario. Il Join Calculus binario e' un linguaggio di programmazione concorrente i cui programmi sono insiemi di processi ognuno specificato da un join pattern binario. Un join pattern binario ha la seguente forma: a() & b() -> c() & d() dove a,b,c,d (non necessariamente distinti) sono messaggi. La semantica e' la seguente: il processo specificato da a() & b() -> c() & d() atomicamente consuma i messaggi a e b ed emette i messaggi c e d, ripetendo tale azione all'infinito. Assumiamo che lo stato iniziale contenga gia' due messaggi “a”.

Esempio1: il programma a() & a() -> b() & c() || b() & c() -> a() & d() || a() & d() -> c() & b() lanciato nello stato iniziale {a,a} dara' luogo alla seguente esecuzione infinita: {a,a} -> {b,c} -> {a,d} -> {b,c} -> {a,d} -> {b,c} -> ...

Esempio 2: il programma a() & a() -> b() & c() || b() & c() -> a() & d() || b() & c() -> b() & d() e' non deterministico. Le due possibili esecuzioni, entrambe finite, sono: {a,a} -> {b,c} -> {a,d} e {a,a} -> {b,c} > {b,d}

Utilizzando primitive di message passing asincrono, scrivere il codice di un server centrale (che gestisca lo stato del sistema) e di un generico processo specificato da a() & b() -> c() & d().

HINT: il processo a() & b() -> c() & d() deve inviare al server una richiesta bloccante di ricezione atomica di a e b, seguita da una spedizione atomica di c e d sempre al server; poi deve semplicemente ciclare.

Esercizio 3: Si considerino il seguente sistema formato da due processi.

```
int x=1;
semaphore S[2] = {2,2};
P_i (i=0,1) {
  for (int j=0; j < 3; j++) {
    P(S[i]);
     if (i==0) { <x+=2;> } else { <x*=2;> }
     V(S[(i+1)%2]);
  }
}
```

a) Ci puo' essere deadlock? Motivare la risposta.

b) Elencare l'insieme di tutti i possibili valori di x al termine dell'esecuzione.
HINT: costruirsi il grafo di tutte le possibili esecuzioni del sistema.

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 2006/2007
PARTE GENERALE - 15 Giugno 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 sistema operativo supporta multithreading secondo il modello **many-to-many**. Vi e' un unico scheduler in **kernel-space** che alterna l'esecuzione di processi secondo la politica round-robin con quanto di tempo di 5ms. Ogni processo implementa in **user space** un proprio scheduler che alterna l'esecuzione dei thread secondo la politica round-robin con quanto di tempo di 2ms. Mostrare il grafo di Gantt ottenuto lanciando i seguenti processi. Le letture e scritture avvengono su un unico device FIFO.

```
process Pi (i=1,2) {
    compute(i*4ms);
    start_threads {
        compute(4ms);
        read(3ms);
        compute(2ms);
    //
        compute(3ms);
        write(2ms);
    }
}
```

Esercizio 2: Consideriamo il seguente stato iniziale per un banchiere multivaluta.

COH (10,15)

	massime	assegnate
P1:	(8,5)	(0,0)
P2:	(5,10)	(0,0)
P3:	(7,7)	(0,0)
P4:	(5,5)	(0,0)

Lo stato e' safe?

Supponiamo che il banchiere riceva la seguente stringa di richieste:
alloc(P1,(3,3)); alloc(P3,(2,3)); alloc(P4,(1,1)); alloc(P1,(5,2)); alloc(P2,(0,7));

Come si comportera' lo scrupoloso banchiere multivaluta di fronte a tale stringa di richieste? Mostrare tutti gli stati intermedi **reali** raggiunti dal sistema e tutti i ragionamenti effettuati dal banchiere. Per ogni stato del sistema non dimenticare di mostrare la coda dei processi sospesi in attesa che la loro richiesta venga esaudita.

Esercizio 3:
Sia **x** l'ultima e **y** la penultima cifra del vostro numero di matricola. Rispondete alla domanda **(y*10+x)%4.**
Elencare vantaggi e svantaggi con particolare attenzione, discutendo le situazioni reali che vi portano gli implementatori di sistemi operativi a compiere determinate scelte.

0. Metti a confronto i differenti tipi di architettura per kernel di sistemi operativi.
1. Metti a confronto due sistemi utilizzabili nel kernel di un sistema operativo per implementare sezioni critiche.
2. Metti a confronto allocazione concatenata e allocazione indicizzata, due tecniche per l'implementazione di file system.
3. Metti a confronto first fit, next fit, best fit, worst fit, alcune politiche per l'allocazione di memoria principale.