# — COMPITO CON SOLUZIONE —

## Sistemi Operativi (modulo II / B)

23 maggio 2013

### Esercizio 1

Si considerino 3 processi P1, P2, P3 che si sincronizzano tramite i semafori contatori S1, S2, inizializzati a 0 e `mutex`, inizializzato a 1:

```
process P1 {
    P(S1);
    P(mutex);
    <A>
    V(mutex);
    V(S2);
}
```

```
process P2 {
    P(mutex);
    <B>
    V(mutex);
    V(S1);
}
```

```
process P3 {
    P(mutex);
    <C>
    V(mutex);
    P(S2);
    <D>
}
```

In quale ordine vengono eseguiti i blocchi di codice A,B,C,D? È possibile che alcuni blocchi vengano eseguiti in parallelo? Elencare tutti i casi possibili spiegando approfonditamente.

I semafori S1 e S2 sono rossi quindi le P sono bloccanti mentre il semaforo mutex è un classico semaforo per la sezione critica inizializzato a 1, che garantisce l'esecuzione di un processo alla volta. Di conseguenza sappiamo che:
- A viene eseguito sempre dopo B (semaforo S1),
- A, B e C non vengono mai eseguiti in parallelo a causa della mutua esclusione,
- D viene eseguito sempre dopo A (semaforo S2), e quindi anche dopo B,
- ovviamente D viene eseguito dopo C perché codice dello stesso processo. Anche se D non è in sezione critica non può quindi essere eseguito in parallelo con gli altri blocchi.

Quindi, i blocchi non sono mai eseguiti in parallelo e otteniamo i seguenti casi:
BACD
BCAD
CBAD

### Esercizio 2

Si considerino `MAX` thread `P(id)` (con `id` $\in [1, \texttt{MAX}]$). Ogni thread calcola una stima della temperatura media su una parte di Italia. I thread invocano la funzione `report` del monitor `previsioni` passando come parametro il proprio `id` e la temperatura calcolata:

```
thread P(id) {
    < computa temperatura >
    previsioni.report(id,temperatura)
}
```

La funzione `report` stampa le temperature passate dai thread in ordine di `id`, **tenendo bloccati i thread** fino al loro turno. Scrivere il codice del monitor `previsioni` illustrandolo approfonditamente.

È sufficiente una variabile intera inizializzata a 1 (current) per sapere quale thread deve stampare e una condition c su cui attendere. Se l'id non è quello corrente il thread attende. Quando un thread stampa incrementa la variabile intera current e fa una notifyAll per permettere agli altri thread in attesa di verificare il proprio id. È necessario l'uso del while per far sì che i thread riverifichino il proprio id ed eventualmente si blocchino di nuovo.

```
monitor previsioni {
  int current=1; // prossimo thread che stampa
  condition c; // condition per l'attesa

  void report(int id, int temperatura) {
    while (id != current)
      c.wait // blocca tutti i thread tranne quello corrente
    print "ID = "+id+", Temperatura = "+temperatura;
    current=current+1; // thread successivo
    c.notifyAll; // notifica tutti in modo che quello corrente possa stampare. Gli altri si ribloccano sul while
  }
}
```