ESERCIZI: SINCRONIZZAZIONE

**ESERCIZIO 1**

Si consideri un sistema costituito da tre processi P1, P2 e P3, e due semafori **generalizzati** S1, S2 inizializzati a zero. I processi hanno la seguente struttura:

|  |  |  |
| --- | --- | --- |
| P1:  repeat  {  V(S1);  }  forever | P2:  repeat  {  P(S1);  print("A");  V(S2);  }  forever | P3:  repeat  {  P(S2);  print("B");  }  forever |

Quale fra le seguenti sequenze non può essere stampata?

1. ABAABAAB
2. ABABABAB
3. AAAABBBB
4. ABAABBBA

Risposta esatta: D

**ESERCIZIO 2**Considerate i seguenti  processi

|  |  |
| --- | --- |
| **Risorse condivise**   **semaphore**  S=**1**, M=**1**; **int** x=10; | |
| **Processo** P1 {     P(M);     x=x+1;     V(M);       P(M);    write(x);     V(M);    V(S); } | **Processo** P2 {     P(S);      P(M);     x=x-1;     V(M);     P(M);    write(x);     V(M); } |

supponete che i processi vengano eseguiti concorrentemente sulla stessa CPU.

1. Individuate le regioni critiche nel codice di P1 e P2.   
   Si possono verificare race conditions per la variabile condivisa x?
2. Determinare tutti i possibili output di tale programma concorrente (P1 in parallelo con P2).
3. Supponiamo di inizializzare il semaforo  **S a 0 invece che a 1**.   
   Determinare tutti i possibili output del programma (P1 in parallelo con P2) modificato.

**Soluzione**

1. Regioni critiche:
   1. R1: x=x+1;
   2. R2: write(x);
   3. R3: x=x-1,
   4. R4: write(x)
2. Le regioni critiche sono protette da semafori per cui non vi sono race conditions su x.
3. Il semaforo S non influisce sul flusso di esecuzione e quindi l'output dipende dall'ordine di esecuzione  
   di R1-R2 e R3-R4:
   1. Prima R1,R3 (in qualsiasi ordine) e poi R2,R4 (in qualsiasi ordine)--> stampa: 10 10
   2. R1 R2 R3 R4 --> stampa: 11 10
   3. R3 R4 R1 R2 --> stampa:  9  10
4. Il semaforo forza l'ordine di esecuzione P1 -> P2 e quindi ci sarà un solo output: 11 10