PCAD a.a. 2017/18 - Scritto del 2 luglio 2018

L'esame è composto da 12 domande a risposta multipla e 1 esercizio a risposta libera. Se richiesto dal testo o se avete dubbi sulla formulazione di una domanda aggiungete una breve spiegazione per giustificare la risposta. Nella stessa domanda ci possono essere da zero a quattro affermazioni vere.
Occorre raggiungere almeno 15 punti per poter far media con il voto della discussione del progetto.

D1 (1 punto) Nei modelli della concorrenza con consistenza sequenziale:
1. Bisogna considerare sono le computazioni con scheduling dei thread basato su strategia round robin
2. Tutti i thread vengono eseguiti almeno per un certo intervallo di tempo
3. Viene garantita l'assenza di deadlock ma non l'assenza di starvation
4. Thread differenti possono condividere stack e heap

D2 (1 punto) Quando si utilizza un semaforo in un programma concorrente
1. i thread non possono accedere simultaneamente alla propria sezione critica
2. l'accesso a dati condivisi da parte di più thread viene serializzato
3. il primo thread che accede al semaforo ha priorità su tutti gli altri nelle istruzioni seguenti
4. è opportuno inizializzare il semaforo a 1 se si vuole puoi usare come mutex

D3 (1 punto) Quando si utilizza un thread pool
1. i thread eseguono task prelevandoli da una coda
2. alla fine dell'esecuzione di un task il corrispondente thread termina
3. non bisogna preoccuparsi della gestione delle race condition nei task
4. non bisogna preoccuparsi dell'allocazione dei task ai thread sottostanti al pool

D4 (1 punto) Un Mutex
1. è un semaforo che può assumere solo i valori 0 e 1
2. è un oggetto immutabile in Java
3. non può essere usato come campo di una struttura dati tipo lista o array
4. può essere implementato disabilitando interruzioni

D5 (1 punto) La tecnica di programmazione lock-free
1. viene usata per minimizzare il numero di istruzioni tra acquire e release di un lock
2. viene usata per rendere efficiente la gestione delle race condition nei programmi concorrenti
3. viene usata per favorire e facilitare l'uso di lock e mutex nei programmi concorrenti
4. viene usata nell'implementazione di strutture dati concorrenti

D6 (1 punto) Una barriera di memoria o memory fence
1. risolve il problema della sezione critica
2. Viene sempre invocata alla fine di blocchi sincronizzati in Java
3. può essere usata per garantire mutua-esclusione in architetture debolmente consistenti
4. ha come effetto quello di disabilitare per più cicli di esecuzione tutte le interruzioni hardware

D7 (2 punti) Il problema della sezione critica
1. si applica a programmi concorrenti qualsiasi
2. richiede di soddisfare le due sole proprietà di mutua esclusione e assenza di deadlock
3. non richiede particolari assunzioni sulla struttura della sezione critica
4. è formulato per programmi concorrenti con al più 2 thread

D8 (2 punti) In un programma concorrente con un input fissato
1. due diverse computazioni non possono eseguire infinite volte la stessa istruzione
2. se una computazione termina allora tutte le possibili computazioni che terminano danno lo stesso risultato
3. se una computazione non termina allora tutte le possibili computazioni non terminano
4. due computazioni diversi non possono dare lo stesso risultato

D9 (2 punti) Nell'esecuzione di un programma concorrente
1. tutti i thread lanciati da un programma vengono sempre eseguiti almeno per un'istruzione
2. i thread vengono eseguiti in parallelo quando possibile ma non necessariamente
3. non è possibile avere due context-switch consecutivi dello stesso thread
4. il numero di context-switch dipende dallo scheduler e dalle operazioni I/O bound dei thread

D10 (2 punti) Quando usiamo oggetti callable in Java
1. Le chiamate dei metodi corrispondenti possono restituire valori
2. Le chiamate dei metodi corrispondenti sono effettuate in mutua esclusione
3. Le chiamate dei metodi corrispondenti sono tutte effettuate in maniera asincrona
4. Non possiamo propagare le eccezioni al di fuori dei metodi corrispondenti

D11 (2 punti) La libreria CyclicBarrier di Java
1. Viene usata per controllare parallelismo tra thread
2. Viene usata come alternativa alle Memory Fence nei modelli con weak consistency
3. Ha un metodo "wait" che viene usato per sincronizzare thread
4. Ha un metodo "notify" che viene usato per sbloccare thread in attesa

D12 (6 punti) Considerate il seguente programma multithreaded MT

```
f=false; g=false;
THREAD P:   while(true) do {f=true; while (!g)  do { print('0'); f=false;} endw; } endw;
THREAD Q:  while(true) do {g=true; while (!f)  do { print('1'); g=false; } endw; } endw;
```

1. Il programma può generare la stringa 01 senza altri output dopo (spiegare risposta)

2. Il programma può generare una stringa infinita di soli 0 (spiegare risposta)

3. Il programma può generare una stringa infinita di soli 01 cioè 010101… (spiegare risposta)

4. Il programma può generare la stringa 100 senza altri output dopo (spiegare risposta)

Esercizio (10 punti)
Partendo dall'algoritmo di Peterson per 2 thread scrivere una sua estensione per risolvere il problema della sezione critica per 4 thread usando uno schema a torneo (semifinali e finale).