PCAD a.a. 2017/18

L'esame è composto da 12 domande a risposta multipla e 1 a risposta libera.
Nelle prime 12 domande bisogna indicare se le affermazioni sono vere o false.
Se avete dubbi sulla formulazione della domanda aggiungete una breve spiegazione per giustificare la risposta. Nella stessa domanda ci possono essere zero o più affermazioni vere.

D1 (1 punto) Nei modelli della concorrenza con consistenza sequenziale:
1. Si considerano sono computazioni con una specifica strategia di scheduling dei thread
2. Tutti i thread vengono eseguiti per un intervallo di tempo prefissato
3. Viene garantita l'assenza di deadlock e starvation e la mutua esclusione
4. Si assume che i thread vengono sempre usati in pool

D2 (1 punto) Quando si utilizza un semaforo in un programma concorrente
1. i thread potrebbero comunque accedere simultaneamente alla propria sezione critica
2. si trasforma un programma concorrente in un programma sequenziale
3. il primo thread che accede al semaforo blocca tutti gli altri thread
4. è opportuno inizializzare il semaforo a 0 se si vuole puoi usare come mutex

D3 (1 punto) Quando si utilizza una barriera di sincronizzazione
1. i thread vengono controllati da un monitor esterno
2. si inseriscono punti di sincronizzazione all'interno dei thread
3. si garantisce la mutua esclusione
4. non bisogna preoccuparsi dell'allocazione dei task ai thread

D4 (1 punto) Una ConcurrentHashMap in Java
1. non può essere usata in blocchi synchronized
2. fornisce metodi thread-safe per inserimento e cancellazione
3. se usata in un programma concorrente garantisce l'assenza di race condition sui propri dati
4. implementa la tecnica dello snapshot per strutture dati concorrenti

D5 (1 punto) Un Reentrant lock
1. è un semaforo binario che viene usato su oggetti con metodi getter e setter
2. viene usato per garantire la mutua esclusione tra thread che aggiornano una variabile condivisa
3. viene usato per evitare deadlock in chiamata ricorsive
4. garantisce starvation-freedom se usato per controllare una risorsa condivisa

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

D7 (2 punti) Il problema della sezione critica
1. si applica a programmi concorrenti con struttura qualsiasi
2. richiede di soddisfare la sola proprietà di mutua esclusione
3. assume che tutti i thread procedano alla stessa velocità
4. è formulato per programmi concorrenti con al più 2 thread

D8 (2 punti) In un programma concorrente:
1. dato un input, esiste una sola computazione possibile
2. dato un input, due diverse computazioni possono dare risultati diversi
3. dato un input, tutte le computazioni terminano oppure tutte le computazioni non terminano
4. dato un input, tutte i possibili scheduling dei thread danno lo stesso output

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 tre 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) Nella libreria RMI
1. L'accesso ad un oggetto remoto è sempre thread-safe
2. Il registry viene gestito dallo stesso server che gestisce un oggetto remoto
3. Il registry serializza le chiamate dei metodi verso un oggetto remoto
4. L'implementazione di un'interfaccia remota deve essere la stessa su server e client

D12 (6 punti) Considerate il seguente programma multithreaded MT

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

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

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

3. Il programma può generare una stringa infinita bababa… (spiegare risposta)

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

Esercizio (10 punti)
Scrivere una possibile implementazione (usando semanfori e variabili condizioni) dell'operazione "wait" usata nelle barriere di sincronizzazione.