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 strategie preemptive
2. Tutti i thread vengono eseguiti almeno una volta
3. Viene garantita l'assenza di deadlock e l'assenza di starvation
4. Thread differenti non possono condividere il proprio stack

Soluzione: F F F V

D2 (1 punto) Quando si utilizza un semaforo in un programma concorrente
1. i thread possono ancora accedere simultaneamente alla propria sezione critica
2. l'accesso a dati condivisi da parte di più thread è sempre serializzato
3. un thread che esegue l'operazione down incrementa il contatore interno al semaforo di uno
4. è opportuno inizializzare il semaforo a 0 se si vuole puoi usare il semaforo come mutex

Soluzione: V F F F

D3 (1 punto) Quando si utilizza un thread pool
1. i thread eseguono task prelevandoli da uno stack
2. alla fine dell'esecuzione di un task il corrispondente thread rimane attivo
3. non bisogna preoccuparsi della gestione della garbage collection della memoria
4. non bisogna preoccuparsi della creazione dei thread

Soluzione: F V F V

D4 (1 punto) Un Semaforo Generale
1. è un semaforo il cui contatore interno può assumere un valore maggiore o uguale a zero
2. è implementato come un oggetto immutabile in Java
3. non può essere usato come campo di una concurrent HashMap
4. contiene al suo interno una coda di thread

Soluzione: V F F V

D5 (1 punto) La tecnica di programmazione basata su confinamento per thread
1. viene usata per ridurre l'uso di lock nei programmi concorrenti
2. viene usata per rendere efficiente la gestione della memoria nei programmi concorrenti
3. viene usata nell'implementazione di server multithreaded
4. viene usata nell'implementazione dei monitor alla Hoare

Soluzione: V F V F

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

Soluzione: F V V F

D7 (2 punti) Il problema della sezione critica
1. si applica a programmi concorrenti qualsiasi
2. richiede di soddisfare almeno le proprietà di mutua esclusione e assenza di starvation
3. non richiede particolari assunzioni sulla struttura della sezione critica
4. è formulato per programmi concorrenti con un numero arbitrario ma finito di thread

Soluzione: F V F V

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

Soluzione: V F F V

D9 (2 punti) Nell'esecuzione di un programma concorrente
1. tutti i thread lanciati da un programma partono sempre simultaneamente
2. i thread vengono eseguiti sempre in parallelo
3. non è possibile alternare context-switch di due diversi thread
4. il numero di context-switch non dipende dallo scheduler

Soluzione: F F F F

D10 (2 punti) Quando usiamo oggetti runnable 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

Soluzione: F F F V

D11 (2 punti) La libreria SynchronizeCollection di Java
1. Viene usata per incapsulare strutture dati per renderle threadsafe
2. Viene usata come alternativa agli ExecutorService
3. Ha un metodo "wait" che viene usato per sincronizzare thread
4. Ha un metodo "notify" che viene usato per sbloccare thread in attesa

Soluzione: V F F F

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 0011 senza altri output dopo (spiegare risposta)

V: P stampa 0 0, Q g=true, P esce ciclo interno, Q stampa 1 1, P e Q solo ciclo esterno con f=g=true

2. Il programma può generare la stringa 1001 senza altri output dopo (spiegare risposta)

V: simile a (1)

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

V: P stampa 0 e poi mette f=false, Q stampa 1 e poi mette g=false, …

4. Il programma può generare la stringa infinita 101 101 … (spiegare risposta)

V: simile a (3)

Esercizio (10 punti)
Partendo dallo schema del Produttore e Consumatore scrivere una algoritmo concorrente per gestire in maniera asincrona la ricerca di una keyword K in un'insieme di pagine web con indirizzi memorizzati in un array URLs di 100 posizioni. Assumere di avere a disposizione una funzione String URLConnection(String A) per scaricare la pagina web all'indirizzo A.

Soluzione

Scheme del produttore e consumatore visto a lezione (es. con mutex e condition o monitor) con buffer circolare dove il produttore inserisce stringhe che rappresentano pagine web ed il produttore legge stringhe e cerca key