NON SPEGNERE IL PC A FINE ESAME

Corso di Sistemi Operativi e Reti

Prova scritta di SETTEMBRE 2018

ISTRUZIONI

1. **Rinomina** la cartella chiamata "CognomeNomeMatricola" che hai trovato sul Desktop e in cui hai trovato questa traccia, sostituendo "Cognome" "Nome" e "Matricola" con i tuoi dati personali;
2. **Carica** tutto il materiale didattico che vorrai usare sul Desktop; puoi farlo solo nei primi 5 minuti della prova;
3. **Svolgi** il compito; lascia tutto il sorgente che hai prodotto nella cartella di cui al punto 1;
4. Quando hai finito lascia la postazione facendo logout,

senza spegnere il PC.

SALVA SPESSO il tuo lavoro

NON SPEGNERE IL PC A FINE ESAME

ESERCIZIO 1 (Programmazione multithread. Punti: 0-20)

Si progetti una struttura dati thread-safe chiamata `BlockingStack`. Tale struttura dati si comporta come una collezione di oggetti con politica di estrazione LIFO, e con metodi di inserimento ed estrazione bloccanti nel caso in cui non siano disponibili rispettivamente posti vuoti o posti pieni. E' inoltre possibile estrarre un elemento qualsiasi posto in qualsiasi posizione nello stack. La dimensione massima della struttura dati è impostabile in fase di creazione di un oggetto di tipo `BlockingStack`.

I metodi pubblici disponibili per la classe `BlockingStack`, operante su oggetti generics di tipo `T` dovranno essere i seguenti:

`void put(T t)` . Aggiunge l'oggetto `t` sullo stack. Si blocca in attesa di slot disponibili qualora lo stack risultasse pieno.

`T take()` . Preleva un elemento dalla cima dello stack e lo restituisce. Si pone in attesa bloccante se non dovessero esserci ancora elementi nello stack.

`T take(T t)`. Estrae l'elemento `t` dallo stack e lo restituisce. Si pone in attesa bloccante qualora l'elemento `t` non dovesse essere ancora presente sullo stack. `t` non deve essere necessariamente posto sulla cima dello stack, ma può trovarsi in qualsiasi posizione.

I metodi richiesti devono essere implementati garantendo la necessaria thread safety; non sono ammesse situazioni di deadlock; è opportuno migliorare l'accessibilità concorrente alla struttura dati ed evitare situazioni di starvation.

NON SPEGNERE IL PC A FINE ESAME

NON SPEGNERE IL PC A FINE ESAME

CI SONO DEI PUNTI AMBIGUI NELLA TRACCIA? COMPLETA TU

È parte integrante di questo esercizio completare le specifiche date nei punti non esplicitamente definiti, introducendo o estendendo tutte le strutture dati laddove si ritenga necessario, e risolvendo eventuali ambiguità.

POSSO CAMBIARE IL PROTOTIPO DEI METODI RICHIESTI? NO

Non è consentito modificare il prototipo dei metodi se questo è stato fornito. Potete aggiungere qualsivoglia campo e metodo di servizio, e qualsivoglia classe ausiliaria, ma NON variare l'interfaccia dei metodi pubblici già specificati.

CHE LINGUAGGIO DEVO USARE? JAVA 7 O SUCCESSIVO

Il linguaggio da utilizzare per l'implementazione è Java. *È* consentito usare qualsiasi funzione di libreria di Java 7 o successivi.

MA IL MAIN() LO DEVO SCRIVERE? E I THREAD DI PROVA? SOLO PER FARE IL TUO DEBUG

Non è esplicitamente richiesto di scrivere un `main()` o di implementare esplicitamente del codice di prova, anche se lo si suggerisce per testare il proprio codice prima della consegna.

NON SPEGNERE IL PC A FINE ESAME

NON SPEGNERE IL PC A FINE ESAME

ESERCIZIO 2 (Linguaggi di scripting. Punti 0-10)

Si scriva uno script in `Perl` dal nome `sql.pl` che offra la possibilità di effettuare semplici query (`join` e `show`) tra tabelle presenti all'interno di un database fornito in input. Nel particolare lo script dovrà operare nel seguente modo:

1. Si lancia lo script passando come argomento il nome di un file di testo contenente un database semplificato

   ```
   ./sql.pl db_filename
   ```

   Il database semplificato contiene solo tabelle a due colonne (un attributo chiave primaria, e un attributo standard). Il file contenente il database semplificato (`database.db` nel nostro esempio) è così composto:
   - Sono presenti `n` righe
   - Ogni riga contiene una tupla *t* così formata: il primo campo indica rispettivamente il nome della tabella di appartenenza di *t*, seguito dal valore della chiave primaria di *t* e il valore del secondo attributo.
     - Esempio di rigo: `(studente,182452,Francesco)`
     - `studente`: Nome tabella
     - `182452`: valore della chiave primaria
     - `Francesco`: valore attributo
2. Il database deve essere indicizzato in memoria utilizzando le dovute strutture dati (a vostra scelta);
3. Lo script rimane in attesa e l'utente può inserire tramite `standard input` una serie di comandi terminati da un fine riga (per semplicità sono richieste solo le operazioni di `join` e `show` successivamente descritte):
   - `join`: effettua il join tra le chiavi primarie di 2 tabelle specificate e mostra il risultato ordinato per chiavi su standard output. Formato del comando:

     ```
     join tableName_2 tableName_2
     ```
   - `show`: mostra il contenuto della tabella scelta ordinando i risultati per per valori di chiave primaria crescenti. Formato del comando:

     ```
     show tableName
     ```
   - `quit`: lo script termina la sua esecuzione
   - **`NON SONO AMMESSI ALTRI COMANDI`**

Attenzione:

Dopo ogni query inviata lo script dovrà mostrarne il risultato.

È possibile lanciare più query durante l'intera esecuzione del programma.

È possibile lanciare una sola query per volta.

Il programma termina **SOLO** quando l'utente inserisce il comando `quit`.

NON SPEGNERE IL PC A FINE ESAME