NON SPEGNERE IL PC A FINE ESAME

Corso di Sistemi Operativi e Reti

Prova scritta di APRILE 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 `BlockingQueuePool` costituita da N differenti BlockingQueue, ciascuna contenente al massimo M elementi di tipo T. Tra le N code costituenti il pool, una è designata come coda corrente per l'operazione di `take()`, e un'altra come coda corrente per le operazioni di `put()`. Le due code correnti possono essere variate e possono eventualmente coincidere; inizialmente la coda designata sia per le operazioni di prelievo che di inserimento è la coda 0. Le BlockingQueue interne al pool non sono direttamente accessibili se non tramite i metodi pubblici di seguito specificati.

`put(T t)`. Pone l'elemento `t` all'interno della coda correntemente designata per l'operazione di inserimento. Si pone in attesa bloccante qualora questa coda fosse piena, inserendo infine `t` non appena si libera un posto nella coda che risulta designata per l'inserimento nel momento in cui è stata effettuata l'operazione di put.

`T take()`. Restituisce un elemento estratto dalla coda correntemente designata per l'operazione di take. Si pone in attesa bloccante qualora questa coda fosse totalmente vuota, restituendo infine un valore non appena entra un elemento nella coda che risulta designata per l'estrazione nel momento in cui è stata effettuata l'operazione di take.

`void nextPut()`. Se la coda correntemente designata per l'inserimento è quella di indice `i`, viene designata la coda di indice `(i+1) % N`. Eventuali operazioni di `put` in attesa bloccante rimangono in attesa sulla coda precedentemente designata per le operazioni di inserimento.

`void nextTake()`. Se la coda correntemente designata per l'estrazione è quella di indice `i`, viene designata la coda di indice `(i+1) % N`. Eventuali operazioni di `take` in attesa bloccante rimangono in attesa sulla coda precedentemente designata per le operazioni di estrazione.

NON SPEGNERE IL PC A FINE ESAME

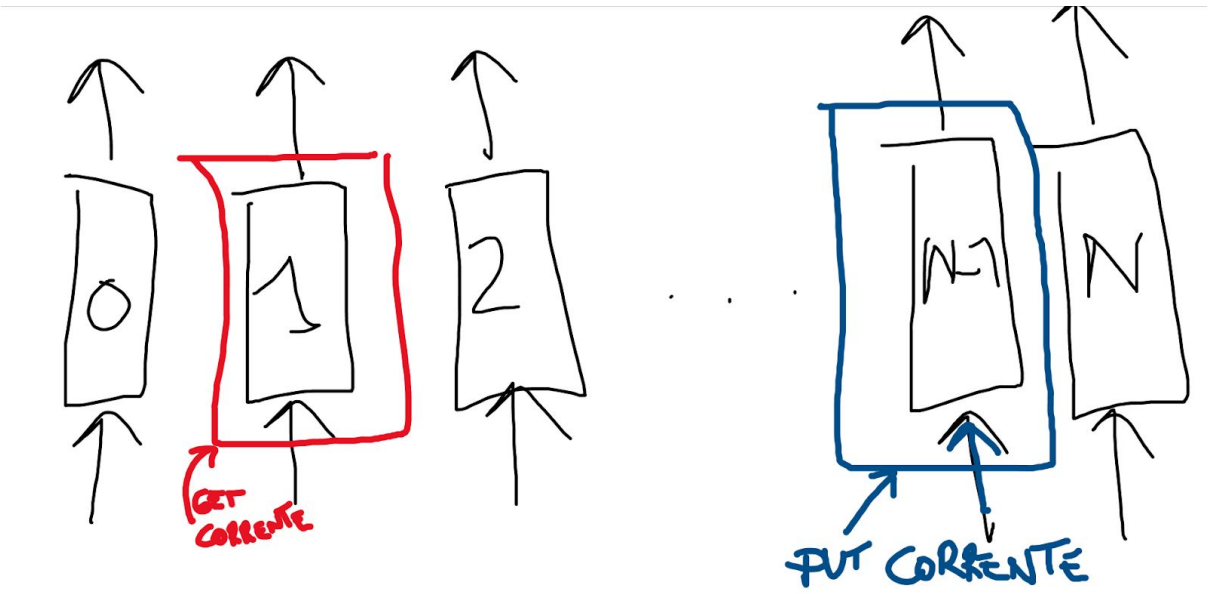

Diagramma esplicativo. Ogni rettangolo rappresenta una coda del pool. In rosso è indicata la coda designata per il prelievo e in blu la coda designata per l'inserimento. Si noti che la coda designata per il prelievo può coincidere in un certo momento con la coda designata per l'inserimento.

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

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

Il file `history.log` contiene una lista di comandi (compresi di eventuali parametri) bash eseguiti precedentemente da un utente tramite il terminale linux.
Un esempio di comando con parametri è il seguente:

```
cp competition.py input.txt /home/francesco/
```

dove:

`cp` è il comando

e

`competition.py input.txt /home/francesco/` sono i parametri

Si scriva uno script perl dal nome `history.pl` che letto il file `history_file` (situtato nella directory corrente) sia in grado di effettuare determinate operazioni.

Lo script dovrà quindi essere invocato con la seguente sintassi:

```
perl history.pl [opzione]
```

Le operazioni che lo script deve effettuare dipendono dall'opzione che l'utente inserisce. Di seguito una breve descrizione di ognuna di esse:

- Se lo script è invocato senza alcuna opzione esso stamperà in output tutti i comandi **con i rispettivi parametri** contenuti all'interno del file `history`.
- Se lo script è invocato con parametro `--sort` esso stamperà in output tutti i comandi **con i rispettivi parametri** ordinati alfabeticamente contenuti all'interno del file `history`.
- Se lo script è invocato con parametro `--stat` esso stamperà in output tutti i comandi invocati seguiti dal numero di volte in cui essi sono stati invocati (anche con parametri diversi)

  Esempio: Il file history.log contiene i seguenti comandi

  ```
  cd ../../
  cd /home/Scrivania
  cd
  ```

  Output:

  ```
  Comando --> Utilizzi
  cd --> 3
  ```
- Se lo script è invocato con parametro `-f (find)` sarà necessario inserire un altro argomento che indicherà quale comando ricercare. Lo script deve quindi filtrare e stampare in output solo il comando ricercato con i suoi parametri.

NON SPEGNERE IL PC A FINE ESAME

Esempio:

File history:

```
cd ../../
cd /home/Scrivania
cd
ls
ls -al
```

Invocazione comando:

```
perl history.pl -f ls
```

Output:

```
ls
ls -al
```

- **TUTTE LE ALTRE OPZIONI NON SONO AMMESSE**