**NON SPEGNERE IL PC A FINE ESAME**

# Corso di Sistemi Operativi e Reti

Prova scritta di SETTEMBRE 2019

PRIMA PARTE

## ISTRUZIONI

1. **Rinomina** la cartella chiamata "Cognome-Nome-Matricola" che hai trovato sul Desktop e in cui hai trovato questa traccia, sostituendo "Cognome" "Nome" e "Matricola" con i tuoi dati personali e **lasciando i trattini**; se hai un doppio nome oppure un doppio cognome dovrai chiamare la cartella come in questo esempio:
    a. DeLuca-MarcoGiovanni-199999
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 deve progettare una struttura dati thread-safe, simile alla Blocking Queue e detta PivotBlockingQueue. In particolare il metodo take() opera su due elementi. Uno, detto elemento PIVOT, deve essere cancellato dalla coda, mentre l'altro deve essere estratto secondo l'usuale politica FIFO. La PivotBlockingQueue può contenere al massimo N elementi, dove N è specificato in fase di creazione della struttura dati. L'elemento PIVOT viene scelto secondo un particolare criterio che è possibile impostare con un metodo apposito.

I metodi di cui deve essere dotata la struttura dati sono:

```
take() -> int
```

individua l'elemento PIVOT e lo elimina dalla coda; quindi estrae e restituisce un elemento secondo il consueto ordine FIFO. Il metodo si pone in attesa bloccante se non sono presenti nella coda almeno due elementi.

```
put(T : int)
```

inserisce l'elemento T nella Blocking Queue. Se la coda contiene già N elementi, individua ed elimina l'elemento PIVOT, quindi inserisce subito l'elemento T.

```
setCriterioPivot(minMax : boolean)
```

Definisce il criterio di scelta dell'elemento PIVOT. Il criterio di scelta dell'elemento PIVOT serve a definire se quando si deve individuare l'elemento PIVOT si deve prendere il massimo oppure il minimo tra gli elementi attualmente presenti nella coda.

Se `minMax = True`, al termine della chiamata il criterio di scelta dell'elemento PIVOT diventerà quello del minimo elemento tra quelli presenti nella coda. Se `minMax = False`, al termine della chiamata il criterio di scelta dell'elemento PIVOT diventerà quello del massimo elemento tra quelli presenti. Se ci sono più di un valore massimo (o più di un valore minimo), deve essere selezionato l'elemento inserito più recentemente. Inizialmente il criterio di scelta dell'elemento PIVOT deve essere impostato su quello del minimo elemento.

**Le strutture dati devono essere implementate garantendo la necessaria thread safety**; non è ammesso progettare strutture dati con potenziali situazioni di deadlock; è opportuno

NON SPEGNERE IL PC A FINE ESAME

migliorare l’accessibilità concorrente alle strutture dati ed evitare, se presenti, situazioni di starvation.

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 POSSO USARE? PYTHON 3.X; oppure JAVA 7 o successivo**

Il linguaggio da utilizzare per l'implementazione è Python 3, o in alternativa, Java. È consentito usare qualsiasi funzione di libreria di Python 3.X o di Java 7 e versioni successive.

**MA IL MAIN() LO DEVO SCRIVERE? E I THREAD DI PROVA? SI**

Sebbene non saranno oggetto di valutazione, è obbligatorio scrivere un `main()` e implementare esplicitamente del codice di prova, che è comunque necessario per testare il proprio codice prima della consegna.

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

Il file */proc/cpuinfo* è un breve file di testo di sola lettura che contiene informazioni riguardanti CPUs (central processing units) di un computer. In particolare, per ogni core del processore sono riportati informazioni quali, ad esempio, `vendor_id`, `model name`, `cpu MHz`, `ecc`. Quindi, se un computer contiene due o più CPUs, le informazioni ad esse relative saranno separate da linee vuote.

Similmente, il file */proc/meminfo* contiene importanti informazioni riguardanti la memoria (`MemTotal`, `MemFree`, `MemAvailable`, `ecc`).

Si scriva uno script `Perl` dal nome `monitor.pl` in grado di monitorare l'utilizzo di **`cpu`** e/o **`memoria`** a seconda delle richieste dell'utente.
In particolare, lo script riceve in input **3 parametri**:

1. cosa monitorare (`all | cpu | mem`). Se si riceve come argomento `all` bisognerà monitorare sia la frequenza della `cpu` che la `memoria` libera; se si riceve tra gli a argomenti `cpu` **oppure** `mem` bisognerà monitorare, rispettivamente, solo la `cpu` o solo la `memoria`.
2. un valore numerico che esprime un determinato `tempo` di attesa in secondi;
3. un numero massimo di `iterazioni` da dover eseguire.

Tutti i parametri sono obbligatori.

Di seguito è riportata la sinossi del comando da implementare

```
./monitor.pl [all|cpu|mem] [time] [iterations]
```

Nello specifico, si vuole monitorare la **frequenza** del clock delle CPUs (si veda il campo `cpu MHz` del file */proc/cpuinfo*) e la **memoria disponibile** ad ogni iterazione (si veda il campo `MemFree` del file */proc/meminfo*).

Lo script dovrà operare nel seguente modo:

1. Si controllano i parametri in input:
    a. Se la prima opzione è `cpu` si legge il file */proc/cpuinfo*, se ne analizza il contenuto, e per ogni processore si stampa su `STDOUT` il numero del processore (`processor`) seguito dal sua frequenza di clock in `MHz` (`cpu MHz`).
    b. Se la prima opzione è `mem` si legge il file */proc/meminfo*, se ne analizza il contenuto, e si stampa su `STDOUT` la quantità di memoria libera in quel determinato istante di tempo (`MemFree`).
    c. Se la prima opzione è `all` si tiene traccia sia della frequenza di clock dei processori (come punta **a**) che della quantità di memoria utilizzata (come punto **b**)
2. Lo script va in pausa per `time` secondi (è possibile utilizzare la funzione `sleep(t)` definita in `Perl`)
3. Si ripulisce la console dalle precedenti stampe (si utilizzi la funzione `reset` —> `print `reset`;`)
4. Si ripetono i punti **1**, **2** e **3** `iterations` volte

NON SPEGNERE IL PC A FINE ESAME

5. Prima di terminare, lo script stampa su un file dal nome `stats.txt` un report in cui sono riportate delle statistiche sull'uso della cpu e/o memoria. In particolare, bisognerà stampare, in ordine decrescente la media della frequenza del clock dei vari processori (se sono presenti più di una CPU) e il massimo valore ottenuto dalla variabile `MemFree` durante tutte le iterazioni

NON SPEGNERE IL PC A FINE ESAME