Corso di Sistemi Operativi e Reti

Corso di Sistemi Operativi

Prova scritta 28 Giugno 2022

ISTRUZIONI PER CHI È IN PRESENZA:

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

ISTRUZIONI PER CHI SI TROVA ONLINE:

1. **Questo file contiene il testo che ti è stato dato ieri, incluso il codice;**
2. **Mantieni a tutto schermo** questo file per tutta la durata della prova; puoi scorrere liberamente tra le sue pagine, ma non puoi cambiare applicazione;
3. **Firma** preliminarmente il foglio che userai per la consegna con nome cognome e matricola;
4. **Svolgi** il compito; puoi usare solo carta, penna e il tuo cervello;
5. **Aiutati** con i numeri di linea per indicare le eventuali modifiche che vorresti fare al codice che ti è stato dato.
6. **Alla scadenza** termina *immediatamente* di scrivere, e attendi di essere chiamato, pena l'esclusione dalla prova;
7. **Quando è il tuo turno** mostra il foglio ben visibile in webcam, e poi metti una foto dello stesso foglio in una chat privata Microsoft Teams con il prof.

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 nuove strutture dati, o estendendo quelle preesistenti laddove si ritenga necessario, risolvendo eventuali ambiguità. Si può cambiare il codice dei metodi esistenti dove serve.

POSSO CAMBIARE IL PROTOTIPO DEI METODI RICHIESTI O DI QUELLI ESISTENTI? 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. Analogamente, i metodi esistenti possono essere modificati nel loro codice, ma non se ne deve cambiare il risultato finale o il significato.

CHE LINGUAGGIO POSSO USARE? PYTHON 3.X

Il linguaggio da utilizzare per l'implementazione è Python 3.6 o successivo. Ricorda che l'operatore di formattazione `f` (esempio, `f"Ciao sono la stringa {testo}"`) è disponibile solo dalla versione 3.6 di Python in poi, ma può essere sostituito con `"Ciao sono la stringa %s" % testo`

POSSO CONSENTIRE SITUAZIONI DI RACE CONDITION NEL MIO CODICE? NO
POSSO CONSENTIRE SITUAZIONI DI DEADLOCK NEL MIO CODICE? NO
POSSO CONSENTIRE ALTRE SITUAZIONI DI BLOCCO TOTALE NEL MIO CODICE, TIPO NESTED LOCKOUT, LIVELOCK O ALTRO? NO
POSSO CONSENTIRE SITUAZIONI DI STARVATION NEL MIO CODICE? SI, tranne quando ti viene chiesto esplicitamente di rimuoverle

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

E' obbligatorio implementare esplicitamente del codice di prova oppure modificare il codice di prova pre-esistente, e accertarsi che giri senza errori prima della consegna.

ESERCIZIO 1 - PROGRAMMAZIONE MULTITHREADED

Punto 1

Si deve modificare la classe `PivotBlockingQueue` in maniera tale da poter specificare quanti pivot ci sono in ciascun momento. Per realizzare questo scopo, introduci il metodo `setPivotNumber(self, n : int)`, che imposta il numero di pivot al valore n (n deve essere almeno 1 e al massimo N-1, dove N è la dimensione della coda).

Il numero di pivot impostato in un certo momento influenza il numero di rimozioni di elementi prescritte. Dovrai fare in modo che il cambio della quantità di pivot influenzi i metodi take() e put() per come spiegato di seguito. Facciamo un esempio e supponiamo che il numero di pivot venga impostato a tre, anziché uno come nel codice preesistente.
Se il numero di pivot è pari a tre, dovrai fare in modo che siano disponibili almeno tre posti liberi prima di inserire l'elemento T. Per fare questo, `put(self, T)` può eliminare *fino a tre* pivot, e cioè *il primo massimo/minimo, il secondo massimo/minimo, e il terzo massimo/minimo*. `take()` resta invece invariata e rimuove un solo pivot per come già stabilito.

E' a tuo carico stabilire come gestire opportunamente il caso in cui ci sono due o più pivot di valore uguale e cioè due o più massimi/minimi di pari valore.

Punto 2

Si estenda la classe `PivotBlockingQueue` con il metodo `putIntegers(...)`. Tale metodo riceve come parametro un numero arbitrario di elementi in coda, ma si blocca in attesa fintantochè in coda non si possa far posto per inserire tutti gli elementi richiesti, eventualmente rimuovendo una parte o tutti gli n pivot, se questi ultimi possono essere utili a far spazio. E' compito tuo stabilire il tipo di dato migliore per codificare gli elementi da inserire. Nota che per risolvere questo punto è necessario aver implementato il punto 1.

Punto 3

Introduci un metodo `waitFor(self, n)` che va in attesa bloccante finché qualche altro thread non introduca nella coda un elemento che supera il valore `n` specificato, , uscendo se la condizione è invece verificata.

SALVA SPESSO

ESERCIZIO 2, TURNO 1 - PERL

Si scriva uno script Perl dal nome **`documentazione.pl`** che estragga alcune informazioni utili dalla documentazione di un certo comando shell ricevuto come parametro.
Lo script deve essere invocato nel seguente modo

```
./documentazione.pl OPZIONE comando
```

I parametri `OPZIONE` e `comando` sono entrambi obbligatori. `comando` rappresenta il nome di un comando disponibile nella proprio shell.

Assumeremo che `comando` abbia delle opzioni **brevi**, le quali iniziano con un solo trattino "-", mentre considereremo come **lunghe** tutte le opzioni che iniziano con un doppio trattino ("--").

I possibili valori del parametro `OPZIONE` sono i seguenti:

- **`-n`**
  quando viene specificata questa opzione bisogna produrre in `STDOUT` il numero di opzioni **lunghe** del comando specificato dal parametro `comando`
  `Esempio:`
  Lo script potrà essere lanciato nel seguente modo:

  ```
  ./documentazione.pl -n rm
  ```

  e dovrà produrre in `STDOUT` la stringa:

10 opzioni lunghe

- **-o**

  Se viene specificata questa opzione, bisogna stampare su un FILE dal nome **opzioni_lunghe.log** l'elenco delle opzioni **lunghe** con relativa descrizione. La stampa dovrà essere fatta ordinando alfabeticamente il nome delle opzioni del comando.

  Esempio:

  Lo script potrà essere lanciato nel seguente modo:

  ```
  ./documentazione.pl -o rm
  ```

  e dovrà produrre in output:

  ```
  --dir: remove empty directories
  --force: ignore nonexistent files and arguments, never prompt
  --help: help display this help and exit
  --interactive[=WHEN]: prompt according to WHEN: never, once (-I), or always (-i); without WHEN, prompt always
  --no-preserve-root: do not treat '/' specially
  --one-file-system: when removing a hierarchy recursively, skip any directory that is on a file system different from that of the corresponding command line argument
  --preserve-root[=all]: do not remove '/' (default); with 'all', reject any command line argument on a separate device from its parent
  --recursive: remove directories and their contents recursively
  --verbose: explain what is being done
  --version: output version information and exit
  ```