

# NON SPEGNERE IL PC A FINE ESAME

## Corso di Sistemi Operativi e Reti

Prova scritta di GIUGNO 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 `DischiConcentrici`. Tale struttura dati simula il comportamento di due array circolari (detti "dischi"), che chiameremo `In` e `Out`, composti da  $N$  valori interi. Ciascun elemento di `In` ha un suo omologo in `Out`. Le coppie di elementi omologhi sono inizialmente impostate considerando gli elementi di `In` e `Out` di pari indice (e cioè, per un dato valore  $i$ , l'omologo di `In[i]` è proprio `Out[i]`), ma è possibile che questa configurazione venga variata ruotando virtualmente `Out` rispetto ad `In`. Ad esempio, ruotando `Out` di 3 posizioni in avanti, avremmo che `In[i]` diventa omologo rispetto all'elemento `Out[(i+3)%N]`.

Nel seguito, dato  $i$ , inteso come indice da applicare sul vettore `In`, chiameremo  $Om(i)$  l'indice omologo da applicare sul vettore `Out`.

Le operazioni che si devono poter compiere su una istanza di `DischiConcentrici` sono:

`shift(m)`. Sposta il disco `Out` di  $m$  posizioni in avanti (o indietro se  $m$  è negativo), aggiornando dunque la corrispondenza tra indici omologhi in accordo.

`set(i, v, d)`. Se  $d=1$ , imposta l'elemento  $i$ -esimo di `In` a  $v$ . Se  $d=0$ , imposta l'elemento attualmente omologo dell'indice  $i$  in `Out` al valore  $v$ . Se a seguito di questa operazione dovesse risultare `In[i] == Out[om(i)]`, bisognerà porre `In[i] = Out[om(i)] = 0`.

`get(i, d)`. Se  $d=1$ , restituisce il valore di `In[i]`. Se  $d=0$ , restituisce il valore di `Out[om(i)]`. Se però,  $d=1$  e `In[i]` dovesse risultare pari a 0, oppure  $d=0$  e `Out[om(i)]` dovesse risultare pari a 0, ci si pone in attesa bloccante fino a che il valore corrispondente non diventa diverso da 0, restituendo infine il nuovo valore.

I tre metodi 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

## 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)

Si scriva uno script in Perl dal nome `search.pl` in grado di localizzare all'interno del filesystem un particolare file/cartella.

```
./search.pl [options] [folder|filename]
```

Lo script può ricevere diversi argomenti/parametri al momento della sua esecuzione ed in particolare dovrà poi svolgere le seguenti operazioni:

- A. Nel caso in cui l'utente inserisca l'opzione `--sync`, lo script dovrà leggere un ulteriore parametro che sarà il path ad una directory del computer locale. Successivamente lo script dovrà essere in grado di individuare tutti i file e le cartelle (anche sottocartelle) presenti nel path specificato e aggiornare (o creare se non esiste ancora) il file `database.db`. Quest'ultimo sarà così composto:

```
path/cartella/in/input --> nome_file_1  
path/cartella/in/input --> nome_file_2
```

**Nota bene:** Il file `database.db` non va mai completamente cancellato e riscritto, ma **SOLO AGGIORNATO**.

**Hint:** È possibile usare il comando `[ls -aR]` per salvare la lista di tutti i file contenuti in una cartella in maniera ricorsiva.

### Esempio di esecuzione 1:

```
1. ./search.pl --sync /home/francesco/Scrivania
```

### Contenuto del file `database.db` dopo esecuzione 1:

```
/home/francesco/Scrivania --> documento1.txt  
/home/francesco/Scrivania --> cartella1  
/home/francesco/Scrivania/cartella1 --> documento2.txt
```

### Esempio di esecuzione 2:

```
2. ./search.pl --sync /home/francesco/Scaricati
```

### Contenuto del file `database.db` dopo esecuzione 2:

```
/home/francesco/Scrivania --> documento1.txt  
/home/francesco/Scrivania --> cartella1  
/home/francesco/Scrivania/cartella1 --> documento2.txt  
/home/francesco/Scaricati --> image1.png
```

# NON SPEGNERE IL PC A FINE ESAME

Come si può notare è stata inserita una sola linea (contenuto della cartella `Scaricati`) senza eliminare le precedenti voci presenti nel database. Nel caso in cui dalla cartella `Scrivania` (o qualsiasi altra cartella) vengano cancellati 1 o più documenti, questi saranno rimossi anche dal database alla successiva sincronizzazione della stessa.

- B. Nel caso in cui l'utente inserisca come argomento direttamente il nome di un file, lo script entrerà in modalità ricerca. In particolare, il file `database.db` dovrà essere letto dallo script e bisognerà stampare il path del file ricercato su `STDOUT`. Nel caso in cui il file non sia indicizzato (non sia presente) nel database, lo script stamperà la stringa `"Nessuna corrispondenza trovata !"` e lo script terminerà.

## Esempio:

### **database.db**

```
/home/francesco/Scrivania --> documento1.txt  
/home/francesco/Scrivania --> cartella1  
/home/francesco/Scrivania/cartella1 --> documento2.txt  
/home/francesco/Scaricati --> image1.png
```

```
./search.pl documento
```

### **Output:**

```
/home/francesco/Scrivania/documento1.txt  
/home/francesco/Scrivania/cartella1/documento2.txt
```

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

**NON SPEGNERE IL PC A FINE ESAME**