

**NON SPEGNERE IL PC A FINE ESAME**

## Corso di Sistemi Operativi e Reti

Prova scritta di FEBBRAIO 2019

### 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)

Bisogna realizzare un sistema di gestione file di attesa presso una certa sede INPS. Una Sede è composta da N uffici distinti, ciascuno identificato da una lettera progressiva (Ufficio 'A', Ufficio 'B', ...). Ciascun ufficio è dotato di una propria coda d'attesa gestita con dei ticket di prenotazione progressivi. Ad esempio i ticket rilasciati dall'ufficio B saranno nel formato "B001", "B002", ecc.

I ticket di prenotazione sono prelevabili dagli utenti degli uffici. Gli addetti degli uffici stessi servono i rispettivi clienti chiamando gli utenti in base all'ordine di erogazione dei ticket di prenotazione; un display visualizza i codici degli ultimi 5 ticket che sono stati chiamati, indipendentemente dall'ufficio di appartenenza. Ad esempio sul display potrebbe essere visualizzata la sequenza:

B004

C076

B005

B006

D113

I metodi che una Sede deve obbligatoriamente fornire sono:

`prendiTicket(uff)`. Rilascia il prossimo ticket disponibile per l'ufficio `uff`. Restituisce una stringa contenente il codice del ticket rilasciato.

`chiamaTicket(uff)`. Chiama il prossimo ticket non ancora servito per l'ufficio `uff`. Ciò che viene visualizzato sul display deve essere aggiornato in accordo. Si pone in attesa bloccante se il prossimo ticket da chiamare non è stato ancora rilasciato.

`waitForTicket(ticket)`. Si mette in attesa che il ticket identificato da `ticket` venga chiamato. Esce quando `ticket` viene chiamato oppure se il ticket risulta tra gli ultimi 5 chiamati e visualizzati sul display.

`printAttese()`. Interrompe la visualizzazione corrente del display mostrando a video il numero di utenti in attesa per ciascun ufficio.

Il display deve essere gestito da un opportuno thread **separato** che aggiorna le informazioni a video quando necessario. ***E' vietato effettuare operazioni di stampa a video direttamente nel corpo dei quattro metodi di cui sopra.***

NOTA: è possibile usare il comando `os.system('cls')` per pulire lo schermo.

**NON SPEGNERE IL PC A FINE ESAME**

Tutti i metodi richiesti per devono essere implementati garantendo la necessaria thread safety; non sono ammesse situazioni di deadlock; è opportuno migliorare l'accessibilità concorrente alle strutture dati ed evitare, se presenti, 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 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.

**NON SPEGNERE IL PC A FINE ESAME**

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

Si scriva uno script Perl dal nome `totobet.pl` in grado di verificare la correttezza delle giocate di una o più *schedine*. Ogni schedina è composta da un gruppo di scommesse calcistiche. In particolare lo script dovrà funzionare nel seguente modo: una volta immessi come argomenti iniziali i nomi di 2 file (rispettivamente il nome del file delle **scommesse** e il nome del file dei risultati delle **partite** giocate) lo script verificherà i risultati delle schedine giocate; calolerà il valore del **moltiplicatore** e della **possibile vincita** e infine verificherà se ogni scommessa è stata **vinta** o meno.

Il valore del **moltiplicatore** si ottiene moltiplicando, per ogni schedina, il valore di tutte le quote relative ai match sui quali si sta scommettendo secondo la formula:

$$\text{moltiplicatore} = \text{quota}_1 * \text{quota}_2 * \dots * \text{quota}_n$$

Il valore della **possibile vincita** è ottenuto moltiplicando l'importo scommesso per il valore del moltiplicatore secondo la formula:

$$\text{possibileVincita} = \text{moltiplicatore} * \text{importoScommesso}$$

Un file delle scommesse è così formato:

```
#Schedina 1#
Roma-Torino          X          1.5
...
Spal-Bologna         X          2.0
#Importo Scommesso   10
#Schedina 2#
...
..
#Schedina n#
Juventus-Chievo      1          1.5
...
Frosinone-Atalanta 2          2.5
#Importo Scommesso   5
```

# NON SPEGNERE IL PC A FINE ESAME

Ogni schedina è composta da più giocate. La prima colonna indica univocamente il match sul quale si sta scommettendo; la seconda colonna mostra il pronostico del risultato dell'incontro ("1" → vittoria della squadra in casa, "2" → vittoria della squadra ospite, "X" → pareggio) mentre la terza e ultima colonna identifica la **quota** della singola giocata.

La squadra di casa è convenzionalmente la prima elencata nel match, mentre la squadra ospite è la seconda squadra. Ad esempio, in Juventus-Chievo, "Juventus" è la squadra di casa e "Chievo" è la squadra ospite.

Il file contenente i risultati finali delle partite/match è così composto:

Roma-Torino	3-2
Udinese-Parma	1-2
Inter-Sassuolo	0-0
Frosinone-Atalanta	0-5
...	
Torino-Udinese	1-0
Sassuolo-Juventus	0-3
Milan-Cagliari	3-0

La prima colonna indica le squadre in competizione mentre la seconda ed ultima colonna indica il risultato ottenuto.

Una volta terminata la procedura di verifica, lo script stamperà su un nuovo file dal nome "check" i risultati ottenuti come nell'esempio di cui sotto.

# NON SPEGNERE IL PC A FINE ESAME

## Esempio file check:

### #Schedina 1#

Roma-Torino	X	1.5 --> NO
Udinese-Parma	1	2.0 --> NO
Inter-Sassuolo	X	3.0 --> OK
Frosinone-Atalanta	2	1.5 --> OK
Spal-Bologna	X	2.0 --> OK

#Importo Scommesso: 10

#Moltiplicatore: 27

#Possibile Vincita: 270

#Vincita: NO

### #Schedina 2#

Juventus-Chievo	1	1.5 --> OK
Genoa-Milan	2	2.0 --> OK
Napoli-Lazio	1	1.5 --> OK
Frosinone-Atalanta	2	2.5 --> OK

#Importo Scommesso: 5

#Moltiplicatore: 11.25

#Possibile Vincita: 56.25

#Vincita: SI