

## **Corso di Sistemi Operativi e Reti, corso di Sistemi Operativi**

**Prova scritta di Settembre 2016**

### **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, ma **senza spegnere** il PC.

### **5. SALVA SPESSO il tuo lavoro**

**PER GLI STUDENTI DI SISTEMI OPERATIVI:** si può sostenere solo uno dei due esercizi se si è già superata in un appello precedente la corrispondente prova. Il tempo a disposizione in questo caso è di 2 ORE.

## ESERCIZIO 1 (Programmazione multithread. Punti: 0-20)

Una `StazioneDiRicarica` per bici elettriche dispone di  $N$  posti ciascuno dei quali può ospitare una `BiciElettrica` a batteria. Una bici elettrica ha un suo livello di carica (0=completamente scarica, 100=completamente carica), e un numero di cicli di ricarica effettuati nell'arco della vita utile della bici stessa (esempio, 0=bicicletta mai usata, 5=bicicletta che è stata ricaricata per cinque volte).



La stazione di ricarica gestisce due tipi di operazioni che possono avvenire in contemporanea e dunque devono essere thread-safe:

1. Richieste di noleggio di biciclette: una richiesta di noleggio di  $X$  biciclette (fino a  $N$ ), si conclude in due modi possibili; o con l'allocazione di  $X$  biciclette, scelte tra quelle più cariche e, a parità di carica, con meno cicli di utilizzo; oppure immediatamente con un errore, se non dovessero essere disponibili tutte le  $X$  biciclette all'atto della richiesta. Nel fornire le biciclette richieste, i posti corrispondenti diventano liberi e a disposizione per il parcheggio di altre bici. Prototipo del metodo corrispondente:

```
ArrayDiBiciclette getBiciclette(int N)
```

Dove `ArrayDiBiciclette` è una struttura dati che può essere scelta dallo studente volta a modellare una collezione di Bici Elettriche. Il valore restituito è `null` in caso di errore o di non disponibilità di tutte le biciclette.

2. Restituzione di biciclette: viene parcheggiato sulla stazione un gruppo di  $X$  biciclette. In questo caso, se non ci sono  $X$  posti disponibili nella stazione bisogna aspettare fino a che non ci sia posto per tutte le bici. Per ogni bici restituita, bisogna incrementare di 1 il numero di cicli di ricarica corrispondenti. Nel restituire le bici, i posti corrispondenti devono venire occupati. Prototipo del metodo corrispondente:

```
putBiciclette(ArrayDiBiciclette A)
```

il metodo deve essere bloccante se non dovessero esserci posti disponibili.

**N.B.** La gestione dell'intero ciclo di vita di una bicicletta (carica, scarica, ecc.) non è esplicitamente richiesta.

*E' parte integrante di questo esercizio completare le specifiche date nei punti non esplicitamente definiti, introducendo o estendendo tutte le strutture dati che si ritengano necessarie, e risolvendo eventuali ambiguità. Non è consentito modificare il prototipo dei metodi se questo è stato fornito.*

**Si può svolgere questo esercizio in un qualsiasi linguaggio di programmazione a scelta** dotato di costrutti di supporto alla programmazione multi-threaded (esempio, C++ con libreria JTC, Java). E' consentito usare qualsiasi funzione di libreria di Java 6 o successivi. 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.

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

La directory *pagelle* contiene un numero variabile di file, ciascuno dei quali riporta le valutazioni (sotto forma di voti numerici) attribuite ai giocatori di diverse squadre di calcio, per una stessa giornata di campionato. In particolare ciascun file sintetizza le valutazioni per una specifica giornata di campionato. A tal scopo, un file presente in questa directory contiene un elenco di blocchi del tipo:

```
SQUADRA:nome_squadra1
giocatore1squadra1-voto_giocatore1squadra1
giocatore2 squadra1-voto_giocatore2 squadra1
...
giocatorensquadra1-voto_giocatorensquadra1

SQUADRA:nome_squadra2
giocatore1 squadra2-voto_giocatore1 squadra2
giocatore2 squadra2-voto_giocatore2 squadra2
...
giocatoren squadra2-voto_giocatoren squadra2

SQUADRA:nome_squadra3
giocatore1 squadra3-voto_giocatore1 squadra3
giocatore2 squadra3-voto_giocatore2 squadra3
...
giocatoren squadra3-voto_giocatoren squadra3

...
```

Come si vede, ciascun blocco inizia con l'indicazione del nome della squadra (*nome\_squadra*) presa in esame e poi riporta su ciascuna linea una coppia di informazioni che associa a ciascun giocatore di quella squadra il voto a lui attribuito in quella giornata (le informazioni sul nome del giocatore ed il suo voto sono separate da un trattino '-').

Il file *fantaSquadre* definisce un insieme di formazioni di fantacalcio. In particolare, memorizza blocchi di informazioni del tipo:

```
SQUADRA:fantasquadra1
FORMAZIONE:giocatore1fantasq1,giocatore2fantasq1,giocatore3fantasq1,..., giocatorenfantasq1

SQUADRA: fantasquadra2
FORMAZIONE:giocatore1fantasq2,giocatore2fantasq2,giocatore3fantasq2,..., giocatorenfantasq2

SQUADRA: fantasquadra3
FORMAZIONE:giocatore1fantasq3,giocatore2fantasq3,giocatore3fantasq3,...,giocatoren fantasq3

...
```

Ciascun blocco indica il nome di una *fantasquadra* e poi la relativa formazione scelta (ovvero un elenco di giocatori i cui nomi sono separati da ',').

Si scriva uno script perl che, a partire dai dati forniti, produca in output il **conteggio dei punti conquistati da ciascuna fantasquadra**, tenendo conto dei punteggi attribuiti ai giocatori che costituiscono la sua formazione.

A tal scopo lo script dovrà calcolare il punteggio totale attribuito a ciascun giocatore sulla base dei singoli voti espressi su di lui nelle diverse giornate di campionato (per far ciò è necessario utilizzare le informazioni contenute nella directory *pagelle*). Quindi, il conteggio dei punti conquistati da una fantasquadra sarà calcolato come la somma dei punteggi attribuiti a tutti i giocatori presenti nella sua formazione.