Corso di Sistemi Operativi e Reti, corso di Sistemi Operativi – 24 Aprile 2015

1. PER GLI STUDENTI DI SISTEMI OPERATIVI E RETI: è necessario sostenere e consegnare entrambi gli esercizi. Sarà attribuito un unico voto su tutta la prova.

2. 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.

Troverete sul vostro Desktop una cartella chiamata "CognomeNomeMatricola" che contiene la traccia dell'elaborato ed eventuali altri file utili per lo svolgimento della prova. Ai fini del superamento della prova è indispensabile rinominare tale cartella sostituendo "Cognome" "Nome" e "Matricola" con i vostri dati personali. Ad esempio, uno studente che si chiama Alex Britti ed ha matricola 66052 dovrà rinominare la cartella "CognomeNomeMatricola" in "BrittiAlex66052".

Per il codice Java, si consiglia di raggruppare tutto il proprio codice in un package dal nome "CognomeNomeMatricola", secondo lo schema usato per rinominare la cartella "CognomeNomeMatricola".

Non saranno presi in considerazione file non chiaramente riconducibili al proprio autore. E' possibile caricare qualsiasi tipo di materiale didattico sul desktop nei primi 5 minuti della prova.

Si consiglia di salvare SPESSO il proprio lavoro.

ESERCIZIO 1 (Linguaggi di scripting)

N.B. Si noti che per totalizzare un voto superiore a 24 su questa prova è necessario svolgere sia la prima che la seconda parte.

PRIMA PARTE (MAX 24 PUNTI)

Il file *eventi.txt* contiene un elenco di ricorrenze o eventi da ricordare. Ciascuna ricorrenza è fornita su una linea del file in accordo al seguente formato:

```
gg-mm: descrizione_evento
```

dove `gg-mm` è un formato data che rappresenta il giorno (con due cifre) ed il mese (con due cifre) in cui cade una certa ricorrenza, mentre `descrizione_evento` è una stringa di testo che descrive l'evento da ricordare.

Ad esempio, un estratto del file **eventi.txt** potrebbe essere il seguente:

```
03-05:martha
20-08:compleanno mamma
02-11:anniversario matrimonio
22-11:compleanno Giulio
20-09:Io 20-08:compleanno Luca
20-09:anniversario zio
27-12:compleanno giusy
```

Si implementi un script perl **remindme.pl** che:

- riceve da linea di comando una data nel formato `gg-mm` (ad esempio `20-09`)
- legge il file *eventi.txt* per individuare eventuali eventi che cadono nella data specificata
- stampi su standard output una stringa del tipo
  - *`Il giorno 20-09 ricorda: compleanno Luca, anniversario zio`*

  se in corrispondenza della data indicata sono presenti degli eventi da ricordare, *oppure*
  - *`Non hai eventi memorizzati per la data 20-09`*

  se in corrispondenza della data indicata non sono presenti degli eventi da ricordare.

SECONDA PARTE (BONUS, MAX 6 PUNTI)

Si implementi una ulteriore funzionalità per lo script **remindme.pl** che consenta di invocare lo script da linea di comando senza parametri e faccia uso al suo interno del comando Linux **`date`** per utilizzare la data corrente come data di riferimento per l'individuazione di eventuali eventi memorizzati nel file *eventi.txt*.

ESERCIZIO 2 (Programmazione multithread)

Un tavolo rotondo a cui sostano N *giocatori (numerati da 0 a N-1)* di poker (N>20) è corredato di 3 posacenere che sono posizionati in origine al centro del tavolo. Ogni posacenere può essere posizionato al centro del tavolo oppure di fronte a un certo giocatore.

I *giocatori* fumano periodicamente, ma per poter svolgere questa operazione è necessario che un posacenere sia nelle immediate vicinanze. Un giocatore *i* può fumare solo se un posacenere risulta posizionato di fronte a sé stesso, oppure di fronte a uno dei due giocatori adiacenti. Si assume che i giocatori siano disposti circolarmente attorno al tavolo.

Si progetti la struttura dati *Tavolo* che consenta le operazioni richieste da parte di N Thread giocatori, garantendo che le regole di accesso ai posacenere siano rispettate, ed evitando situazioni di inconsistenza, deadlock e possibile starvation.

***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 dei thread di prova, anche se lo si suggerisce per testare il proprio codice prima della consegna.