Corso di Sistemi Operativi e Reti, corso di Sistemi Operativi – Giugno 2016

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.

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

Non saranno valutate prove d'esame il cui codice è stato messo nel package di default.

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. Punti 0-10)

Si consideri il seguente file *info.txt* che contiene blocchi omogenei di informazioni testuali. Ciascun blocco di informazioni è separato da una riga vuota e contiene i medesimi campi chiave (che nell'esempio sono `Hostname`, `OS`, `MemoryGB`, `CPUcount`, `DiskSpace`, `Location`).

```
Hostname: computer1
OS: Windows 2008 Server
MemoryGB: 8
CPUcount: 2
DiskSpace: 160GB
Location: cubo1

Hostname: computer2
OS: Windows Server 2012
MemoryGB: 16
CPUcount: 4
DiskSpace: 500GB
Location: cubo1

Hostname: computer3
OS: Windows Server 2008
MemoryGB: 16
CPUcount: 2
DiskSpace: 80GB
Location: cubo2

…
```

Si scriva uno script perl per la conversione di un file di testo nel formato sopra indicato in un nuovo file in formato `csv`.

Un file `.csv` definisce una organizzazione tabellare dei dati, in cui ogni riga della tabella è rappresentata da una linea di testo, che a sua volta è divisa in campi separati da un apposito carattere separatore e ciascuno dei campi rappresenta un valore. Nel caso specifico si vuole ottenere un nuovo file *info.csv* in cui la prima riga di testo rappresenta l'intestazione della tabella (ovvero contiene i campi chiave utilizzati nei blocchi del file testuale) e le successive righe indicano i valori contenuti in ciascun blocco. Il carattere separatore da utilizzare è rappresentato dalla virgola.

Nella figura è riportato il risultato atteso dalla conversione del file *info.txt* nel file *info.csv*. Come si vede, la prima riga contiene i campi chiave presenti nei blocchi del file di testo originario, mentre le successive contengono i valori presenti in ciascun blocco (in corrispondenza ai relativi campi chiave), inoltre ciascun campo è separato da virgola.

```
Hostname, OS, MemoryGB, CPUcount, DiskSpace, Location
computer1, Windows 2008 Server, 8, 2, 160GB, cubo1
computer2, Windows Server 2012, 16, 4, 500GB, cubo1
computer3, Windows Server 2008, 16, 2, 80GB, cubo2
…
```

ESERCIZIO 2 (Programmazione multithread. Punti: 0-20)

Una `Pizzeria` dispone di un numero illimitato di tavoli, ciascuno dei quali dispone di un numero illimitato di posti. Ogni `Tavolo` (detto anche "Comitiva") può avere in un certo momento un determinato numero di pizze servite o ancora da servire.

Una pizza servita è rappresentata da una stringa che può essere una qualsiasi tra "`(*)`", "`(-)`", "`(@)`", "`(O)`" ecc. ecc. Una pizza ancora da servire è rappresentata da una stringa analoga come "[*]", "[-]", "[@]", "[O]", ecc. ecc.

Un tavolo è modellato come una stringa che rappresenta le pizze servite e da servire in un certo momento. Ad esempio

```
T = "(*)[!](@)[@][@]"
```

Indica che nel tavolo T è già stata servita una pizza "(*)" e una "(@)", mentre devono ancora essere servite "[!]" e "[@]" "[@]".

Un `PizzaioloPazzo` cucina e serve a intervalli regolari una pizza di tipo casuale, mentre nuovi tavoli carichi di commensali arrivano in pizzeria generati dal "Dio dell'amicizia" (`FriendshipGod`).

Si progetti una pizzeria dove 4 Camerieri lavorano per servire la pizza ai tavoli sfruttando le pizze preparate dal pizzaiolo pazzo, e facendo accomodare le comitive (dette "Tavoli") mano a mano che queste vengono generate dal Dio dell'amicizia.

I compiti del Cameriere devono essere:

1. Servire le pizze generate dal pizzaiolo pazzo e poggiarle sui tavoli dove tali pizze mancano, facendo attenzione alla corrispondenza del tipo di pizza (esempio, si deve servire "`(*)`" dove si trova un "`[*]`";
2. Gestire le comitive in arrivo prodotte dal Dio dell'amicizia;
3. Eliminare i tavoli dove tutte le pizze sono state servite;

La scelta dell'ordine delle tre operazioni di cui sopra e la strategia di gestione del servizio sono a carico dello studente. Viene fornito il codice del PizzaioloPazzo, del FriendshipGod, di parte della Pizzeria e del Cameriere, che possono essere integrati a piacere e devono essere completati, nel RISPETTO delle specifiche fornite.

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.