Corso di Sistemi Operativi e Reti, corso di Sistemi Operativi – 29 Giugno 2015 – Tempo a disposizione 3.5 ore.

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)

(0-10 Punti). E' dato l'output del comando `conntrack -L` che riporta l'elenco delle conversazioni al momento in corso su un certo firewall (una conversazione per rigo). Tale output è così strutturato (lo studente può trovare nel file "conn.txt", fornito a corredo della traccia, un esempio di prova)

```
tcp      6 431859 ESTABLISHED src=160.97.63.90 dst=193.206.135.89 sport=9080 dport=80 src=193.206.135.89 dst=160.97.63.90 sport=80 dport=9080 [ASSURED] mark=0 use=1
tcp      6 211544 ESTABLISHED src=94.236.48.40 dst=160.97.62.87 sport=39190 dport=35002 [UNREPLIED] src=160.97.62.87 dst=94.236.48.40 sport=35002 dport=39190 mark=0 use=1
tcp      6 209447 ESTABLISHED src=94.236.48.40 dst=160.97.62.66 sport=39190 dport=46987 [UNREPLIED] src=160.97.62.66 dst=94.236.48.40 sport=46987 dport=39190 mark=0 use=1
tcp      6 118 TIME_WAIT src=160.97.62.67 dst=62.67.193.41 sport=65350 dport=80 src=62.67.193.41 dst=160.97.62.67 sport=80 dport=65350 [ASSURED] mark=0 use=1
udp      17 29 src=160.97.62.2 dst=173.245.58.107 sport=36791 dport=53 src=173.245.58.107 dst=160.97.62.2 sport=53 dport=36791 mark=0 use=1
udp      17 66 src=160.97.63.98 dst=74.125.232.150 sport=60829 dport=443 src=74.125.232.150 dst=160.97.63.98 sport=443 dport=60829 [ASSURED] mark=0 use=1
```

dove subito dopo la prima occorrenza delle stringhe "`src=`" e "`dst=`" è possibile trovare gli indirizzi IP rispettivamente di sorgente iniziale e destinazione iniziale del flusso di byte tra stazione client e stazione server.

Si scriva uno script Perl che riceva da standard input un file nel formato di cui sopra, e ritorni su standard output, uno per rigo: l'elenco dei primi dieci indirizzi IP ordinati per numero di conversazioni in cui questi compaiono come destinazione iniziale, insieme a tale numero di conversazioni; l'elenco dei primi dieci indirizzi IP destinazione ordinati per numero di conversazioni in cui compaiono come sorgente iniziale, insieme a tale numero di conversazioni.

(0-5 Punti): si aggiunga allo script precedente l'opzione da riga di comando '`--dns`' che, se presente, impone di visualizzare il nome host della macchina corrispondente a un certo IP, facendo una opportuna risoluzione DNS (se un certo indirizzo IP non è associato ad alcun nome, si stampi l'indirizzo IP stesso).

Un output di esempio può essere:

```
-------TOP DESTINATIONS-------------------------------
IP ADDRESS                                    CONNECTIONS
-----------------------------------------------------
160.97.63.56                                          161
download.eclipse.org                                   87
160.97.63.64                                           80
160.97.62.57                                           76
160.97.63.98                                           75
160.97.63.97                                           74
host196-247-110-95.serverdedicati.aruba.it             71
160.97.63.86                                           64
160.97.62.236                                          62
ml.mat.unical.it                                       51
-------TOP SOURCES--------------------------------------
94.236.48.40                                          975
160.97.63.98                                          472
gr-1-3-0-70.blacklotus.net                            419
79.133.58.90                                          417
67.227.189.125                                        295
160.97.62.67                                          271
160.97.63.99                                          266
ltwinters.safrafinanciamentos.com.br                  261
65.55.54.39                                           242
160.97.63.97                                          240
```

ESERCIZIO 2 (Programmazione multithread)

(0-15 Punti): La struttura dati *Studente* è dotata dei seguenti campi:

1. Un numero di matricola;
2. Un nome e un cognome;
3. Un elenco di *esami* sostenuti;

Un esame è a sua volta costituito da:

1. Un codice dell'insegnamento;
2. Un voto da 18 a 30;
3. Un valore booleano che esprime se è stata attribuita la lode;
4. Una data di sostenimento, espressa come numero di giorni trascorsi dal 1 Gennaio 2000.

La classe *Studente* deve essere dotata dei seguenti metodi, ciascuno dei quali deve essere progettato in maniera tale da essere *Thread-safe* (non devono esserci problemi di inconsistenza se due o più thread invocano contemporaneamente uno dei metodi).

`String getNome()`. Restituisce la concatenazione di nome e cognome;

`void setNome(String n, String c)`. Imposta nome e cognome ai rispettivi nuovi valori parametrici;

`void addEsame(Esame E)`: aggiunge l'esame `E` alla carriera dello studente corrente;

`double getMedia()`: restituisce la media in trentesimi dello studente corrente;

`boolean ripulisci()`: elimina dalla carriera dello studente eventuali esami che dovessero essere stati sostenuti due volte, o per i quali il voto non dovesse essere ricompreso tra 18 e 30, o per i quali è stata attribuita la lode senza che il voto sia 30. Restituisce *true* se viene applicata qualche modifica alla carriera dello studente, *false* altrimenti.

Si tenga conto del fatto che ogni istanza della classe studente può essere usata anche da centinaia di thread in contemporanea, per cui è necessario di ottimizzare le politiche di locking della struttura dati nella maniera più efficiente possibile.

***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.