

Corso di Sistemi Operativi e Reti, corso di Sistemi Operativi – Luglio 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.

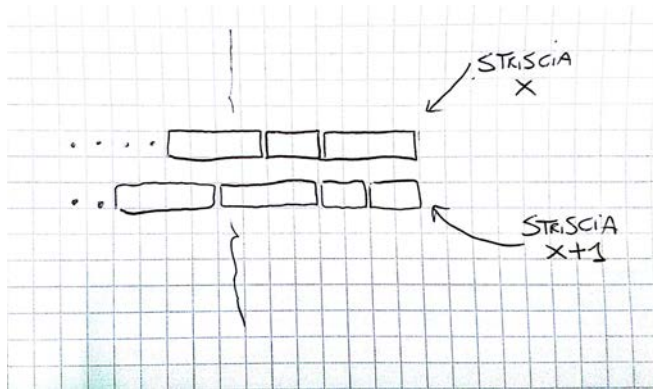
Non saranno presi in considerazione file non chiaramente riconducibili al proprio autore. E' possibile caricare qualsiasi tipo di materiale didattico sul desktop nei primi 5 minuti della prova.

Si consiglia di salvare SPESSO il proprio lavoro.

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

Un Traghetto è composto da 6 strisce di 50 “celle” ciascuna. Il Traghetto è gestito da 4 Parcheggiatori, i quali sistemano in contemporanea le Vetture che via via si presentano all’imbarco. Una Vettura può essere di due tipi: Automobile, di dimensione 1x2 celle, o Autobus, di dimensione 1x4 celle. Le vetture possono essere posizionate nel parcheggio solo in posizione orizzontale all’interno di una striscia, solo se in tale striscia è disponibile un numero sufficiente di celle, e nell’ordine di arrivo nella striscia che viene scelta.

Ad esempio, se la situazione attuale nel traghetto è la seguente:



E ci si trova a dover posizionare un Autobus, lo si potrà sistemare nella Striscia X immediatamente a sinistra dell’ultima vettura posizionata in questa striscia, poiché ci sono 4 celle libere. Invece, se si deve posizionare una Automobile, sarà possibile utilizzare sia la Striscia X che la Striscia X+1, poiché entrambe hanno 2 celle libere.

Una SorgenteMacchine produce delle vetture di tipo casuale da posizionare all’interno del Traghetto.

Si modelli il Traghetto e i Parcheggiatori, in maniera tale che questi ultimi sistemino le vetture prodotte dalla SorgenteMacchine secondo le regole prescritte e con la massima contemporaneità possibile. In particolare è necessario completare il metodo CaricaVetture della classe Traghetto, il quale termina nel momento in cui non è più possibile caricare ulteriori Vetture di alcun tipo.

Viene fornito il codice della SorgenteVetture, di parte del Traghetto e del Parcheggiatore, 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. Il codice prodotto deve essere, in ordine di importanza, thread-safe, esente da possibili deadlock, esente da possibile starvation, e garantire la massima contemporaneità delle operazioni.

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)

Si consideri un file di log *access_log*, che registra tutte le richieste HTTP fatte ad un server. Ciascuna riga del file *access_log* è del tipo:

```
site logName fullName [date:time GMToffset] "req file proto" status length referer user-agent
```

dove **site** rappresenta l'IP o il nome host del client che ha effettuato la richiesta HTTP; *logName* indica il login del client che ha effettuato la richiesta (oppure '-' se non specificato); *fullName* rappresenta il nome del client che ha effettuato la richiesta (oppure '-' se non specificato); *date* è la data della richiesta HTTP; *time* è l'ora della richiesta HTTP; *GMToffset* indica il fuso orario; *req* indica il tipo di richiesta http (ad esempio GET); **file** è il path o il nome del file richiesto; *proto* indica il tipo di protocollo usato per la richiesta (ad esempio "HTTP/1.1"); **status** è il codice di stato HTTP a 3 cifre restituito dal web server rappresentativo dello stato della richiesta; *length* è la lunghezza in byte dell'oggetto richiesto. Infine, gli ultimi due campi *referer* e *user-agent* identificano la pagina che riferisce la risorsa richiesta e lo user agent (nome dell'applicazione, sistema operativo..).

Ad esempio, una riga del file *access_log* potrebbe essere la seguente:

```
61.9.4.61 - - [08/Mar/2004:07:27:37 -0800] "GET /MSOffice/cltreq.asp?UL=1&ACT=4&BUILD=2614&STRMVER=4&CAPREQ=0 HTTP/1.0" 404 284
```

Si scriva uno script perl capace di analizzare il file di log specificato da linea di comando al fine di ricavare alcune informazioni. In particolare, lo script deve produrre dei nuovi file su cui annotare informazioni circa lo stato delle richieste provenienti dai diversi ip (**site**) e in relazione a specifiche risorse (**file**). A tal scopo lo script deve creare in modo automatico dei nuovi file su cui poter scrivere tali informazioni. Ciascuno di questi nuovi file deve chiamarsi come "*stato_status*", dove *status* è il valore numerico che corrisponde ad uno degli stati trovati durante la lettura del file *access_log* (cioè in corrispondenza del campo **status** che si trova su ciascuna riga). Perciò, se ad esempio, nel file di log dovessero essere presenti in totale cinque distinti valori di status (200, 302, 303, 401, 404) lo script dovrà creare altrettanti file di testo (*stato_200*, *stato_302*, *stato_303*, *stato_401*, *stato_404*). **Nota che questo e' un esempio, mentre lo script deve funzionare con file di input dove i codici di stato che compaiono possono essere diversi.**

Lo script, deve quindi :

1. leggere il file *access_log* indicato linea di comando (esempio di invocazione: "mioscript.pl <NOMEFILE>");
2. individuare per ciascuna richiesta (e cioè per ciascuna riga del file) il valore dei campi **site**, **file** e **status** ;
3. sulla base di questi valori individuati deve aggiornare opportunamente il relativo file *stato_status* con le informazioni corrispondenti al **site** ed al **file** corrispondenti

Ad esempio, per le seguenti tre righe di esempio del file *access_log*:

```
61.9.4.61 - - [08/Mar/2004:07:27:37 -0800] "GET /MSOffice/cltreq.asp?UL=1&ACT=4&BUILD=2614&STRMVER=4&CAPREQ=0 HTTP/1.0" 404 284
61.9.4.61 - - [08/Mar/2004:07:27:36 -0800] "GET /_vti_bin/owssvr.dll?UL=1&ACT=4&BUILD=2614&STRMVER=4&CAPREQ=0 HTTP/1.0" 404 284
208.247.148.12 - - [11/Mar/2004:08:14:18 -0800] "GET /mailman/listinfo/ppwc HTTP/1.0" 200 6252
```

lo script deve aggiungere nel file *stato_404* le coppie di valori

```
61.9.4.61 : /MSOffice/cltreq.asp?UL=1&ACT=4&BUILD=2614&STRMVER=4&CAPREQ=0
61.9.4.61 : /_vti_bin/owssvr.dll?UL=1&ACT=4&BUILD=2614&STRMVER=4&CAPREQ=0
```

e nel file *stato_200* la coppia di valori

```
208.247.148.12 : /mailman/listinfo/
```