Corso di Sistemi Operativi e Reti, corso di Sistemi Operativi – Settembre 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 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)

Si implementi uno script perl **checker.pl** per la gestione di una procedura di login ad una applicazione. Lo script deve funzionare come descritto di seguito, in due differenti modalità.

Nota: nel seguito si fa riferimento ad un *file delle credenziali*, necessario per memorizzare le informazioni di login per ciascun utente (costituite dalla coppia <`username`, `password`>). Si precisa che è a carico dello studente la definizione di questo file (scelta del nome del file, del formato di memorizzazione dei dati e suo utilizzo).

Modalità 1 (nuovo utente)

In questa modalità di funzionamento lo script riceve 2 parametri, rispettivamente, una `username` ed una `password`. Quindi:

- verifica che il nome utente indicato come `username` non sia stato già utilizzato. Ciò equivale a verificare che la `username` non sia compresa tra quelle già inserite e opportunamente memorizzate sul *file delle credenziali*. Nel caso in cui il nome sia stato già utilizzato, lo script risponde con un messaggio di errore e termina.
- Se, invece, si tratta di un nuovo nome utente, lo script procede alla verifica della `password` inserita. In particolare, lo script verifica che i criteri di scelta della password siano corretti. Una password, infatti, viene accettata se rispetta i seguenti requisiti:
    - lunghezza superiore a 7,
    - utilizzo congiunto di lettere minuscole e maiuscole
    - utilizzo di almeno uno dei simboli dell'insieme {. , +, - , _ , #}
- Se la password viene accettata perché in accordo ai requisiti lo script memorizza le informazioni di login del nuovo utente, e cioè, la sua `username` e la sua `password` aggiornando opportunamente il *file delle credenziali.*

Modalità 2 (utente già registrato)

In questa modalità di funzionamento lo script riceve 1 solo parametro, corrispondente alla `username` di un utente che si suppone essere stata già inserita. Quindi:

- verifica che il nome utente indicato come `username` sia effettivamente compreso tra quelli previsti. Ciò equivale a verificare che la `username` sia effettivamente presente tra quelle già inserite e opportunamente memorizzate nel *file delle credenziali*.
- Se la `username` non risulta presente tra quelle memorizzate, lo script termina con un messaggio di errore.
- Al contrario, se presente, lo script richiede all'utente l'inserimento della password da tastiera e procede alla *verifica di uguaglianza* della password inserita da tastiera con quella memorizzata sul *file delle credenziali* in corrispondenza della `username` fornita. Se le due `password` risultato identiche, lo script abilita l'utente all'accesso. Altrimenti, dopo un massimo di tre tentativi di inserimento di una password termina fornendo un messaggio di errore.

ESERCIZIO 2 (Programmazione multithread)

(0-15 Punti): Una `Vaschetta` di gelato è modellata attraverso la struttura dati `<codicegusto,quantita>`, dove il valore `quantita` rappresenta un valore intero espresso in grammi nell'intervallo da 0 a 1000.

Una `Gelateria` è costituita da una collezione di vaschette. Le gelaterie servono coni a due gusti in base alle richieste dei clienti, mantenendo aggiornate le quantità disponibili (ogni pallina di gelato pesa circa 45 grammi). E' possibile inoltre riempire fino all'orlo una determinata vaschetta in qualsiasi momento.

Si deve implementare la classe `Gelateria`, corredata dai seguenti metodi thread-safe:

`void getCono(Gusto1, Gusto2)`: preleva una pallina ciascuna rispettivamente dalle vaschette `Gusto1` e `Gusto2`. Se le quantità richieste non dovessero essere disponibili, il thread chiamante viene posto in attesa fino a nuova disponibilità dei gusti richiesti. Si noti che `Gusto1` e `Gusto2` possono coincidere.

`int getConoCumeDede(Gusto1, Gusto2)`: come `getCono`, ma in caso di indisponibilità delle quantità necessarie per uno o tutti e due i gusti preleva arbitrariamente delle palline di altri gusti dalle vaschette che risultano avere sufficiente quantità al proprio interno, senza comportare attese per il thread chiamante.

Il valore restituito deve essere: 1, se il cono corrisponde a quanto richiesto; 0, se uno o tutti e due i gusti sono stati cambiati; -1 se non è stato possibile comporre il cono (non ci sono le quantità sufficienti in nessuna vaschetta).

`void riempi(Gusto1)`: rabbocca fino al valore 1000 la vaschetta identificata da `Gusto1`.

***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.