Corso di Sistemi Operativi e Reti, corso di Sistemi Operativi – 1 Giugno 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)

La directory `programmi` contiene diversi file testuali (ad esempio, `Rai1`, `Rai2`, `Rai3`, `RaiMovie`, `Raisport1`, etc), ciascuno dei quali definisce il palinsesto dei programmi tv per una giornata. Le informazioni in ciascuno dei file sono riportate per linea e, in particolare, ciascuna linea è del formato

```
hh:mm – titolo del programma
```

ad indicare che un certo programma inizia ad una determinata ora.

Il file `info` riporta invece delle informazioni aggiuntive relative al alcuni programmi, film, serie tv. Le informazioni in questo file sono fornite per linea e nel seguente formato

```
titolo del programma===> informazioni testuali
```

Si scriva uno script perl che a partire dalle informazioni (sulla fascia oraria) fornita da linea di comando riporti su standard output l'elenco (e l'eventuale descrizione) dei programmi che sono previsti nel palinsesto per l'intero intervallo orario.

Lo script deve:

- leggere da linea di comando l'informazione sulla fascia oraria indicata, fornita nel formato `hh` (ad esempio `09`, `10`, `21`, `23`, …)
- individuare all'interno della directory `programmi` tutti i programmi che ricadono nella fascia oraria indicata. Ad esempio, se la fascia oraria indicata è `21` devono essere restituiti tutti i programmi previsti nell'intera ora (quindi, ad esempio, quelli programmati per le `21:00`, `21:15`, `21:30`, `21.55` e così via)
- restituire su standard output l'elenco dei programmi individuati e (se presente) anche l'informazione aggiuntiva riportata nel file `info` per quel programma.

ESERCIZIO 2 (Programmazione multithread)

Bisogna progettare una classe chiamata *RandomVector* dotata del metodo

```
Vector getRand()
```

Il metodo ritorna un vettore V di 5 elementi di tipo double, dove:

-V[0] è un numero intero scelto casualmente;
-V[1] è la radice quadrata di V[0];
-V[2] è il quadrato di V[0];
-V[3] è il quadrato della somma di V[1] e V[2];
-V[4] è la radice quadrata della somma di V[1] e V[2].

Il metodo deve essere implementato tenendo conto della disponibilità delle seguenti tipologie di Thread, che devono obbligatoriamente essere implementate e utilizzate:

-Thread di tipo *Sommatore*, l'unico thread all'interno della cui esecuzione è possibile effettuare addizioni;
-Thread di tipo *Sceglitore*, l'unico all'interno della cui esecuzione è possibile invocare operazioni di estrazione di numeri casuali;
-Thread di tipo *SqrtThread*, l'unico all'interno della cui esecuzione è possibile invocare operazioni di estrazione della radice quadrata di un numero;
-Thread di tipo *SqrThread*, l'unico all'interno della cui esecuzione è possibile invocare operazioni di calcolo del quadrato di un numero.

Le tipologie di thread di cui sopra devono essere utilizzate risolvendo tutti le situazioni di possibile assenza di thread-safety, e cercando di ottimizzare l'efficienza complessiva.

***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.