Reti di Calcolatori T
Appello del 08/01/2018
Compito 1

Cognome: ……
Nome: ……
Matricola: ……

Tempo a disposizione: 2h

È obbligatorio inserire Cognome, Nome, e numero di Matricola all'inizio di ogni file sorgente, pena la non valutazione del compito, che viene stampato in modo automatico solo in caso siano presenti gli elementi detti sopra.
Si devono consegnare **tutti i file sorgente e tutti gli eseguibili prodotti singolarmente** (*per favore, solo quelli relativi ai file sorgente consegnati!!!*).

La prova intende valutare le capacità progettuali e di programmazione sia in **ambiente Java** che in **ambiente C**, pertanto è consigliato sviluppare **entrambe** le soluzioni richieste al meglio.
In entrambi gli esercizi, sia in Java che in C, si effettuino gli opportuni controlli sugli argomenti della richiesta e si gestiscano le eccezioni verso l'utente, tenendo presente i criteri secondo cui si possa ripristinare il funzionamento del programma oppure si debba forzarne la terminazione.
Leggete con attenzione le specifiche del problema prima di impegnarvi "a testa bassa" nello sviluppo delle singole parti. Naturalmente, ci aspettiamo che i componenti da consegnare siano stati provati e siano funzionanti.

****

Si richiede il progetto della gestione dei servizi **BefanaFelice**, per la gestione della consegna delle calze della Befana da parte di una associazione di volontariato.
I servizi di BefanaFelice mantengono, per ogni calza, le seguenti informazioni: un **identificatore** del bambino che riceverà la calza, unico all'interno del sistema; il **tipo** di calza che può essere 'Normale' o 'Celiaco'; **carbone**, 'S' (sì) o 'N' (no) che indica la presenza di un pezzo di carbone; **città** e **via** di residenza; **messaggio**, un campo libero di 256 caratteri dove scrivere un breve messaggio.
Si vogliono realizzare le funzionalità di gestione:

1. **inserimento di una nuova calza**: questa operazione richiede *l'identificatore* del bambino, il *tipo*, il *carbone*, la *città*, la *via* e il *messaggio*, ed inserisce la nuova calza nella struttura dati;
2. **visualizzazione di tutte le calze di un certo tipo e carbone**: questa operazione richiede il *tipo di calza* e l'indicazione di *carbone* e restituisce l'elenco (identificatore) di *tutte le calze di quel tipo e con/senza carbone*;
3. **visualizzazione di tutte le città dove saranno consegnate calze**: questa operazione restituisce la lista di *tutte le città dove sarà consegnata almeno una calza*;
4. **visualizzazione del numero di calze consegnate in una città e una via**: questa operazione richiede una *città* e una *via* e restituisce *il numero di calze da consegnare in quella città e via*.

Si progetti con particolare attenzione la **struttura dati** che mantiene lo stato, fino ad un massimo di N calze (L, per libero a default o valore equivalente), da implementare opportunamente nei diversi ambienti richiesti, Java e C.

Identificatore	Tipo	Carbone	Città	Via	Messaggio
L	**L**	**L**	**L**	**L**	**L**
MarioRossi1	**Normale**	**N**	**Bologna**	**Saragozza**	**Bravo Mario!**
MarioBianchi1	**Celiaco**	**S**	**Roma**	**Veneto**	**Mario sei birichino…**
…	**…**	**…**			**…**
MariaRossi12	**Normale**	**S**	**Firenze**	**Larga**	**Maria comportati meglio…**
L	**L**	**L**	**L**	**L**	**-1:-1**

Si considerino e si segnalino le possibilità di interferenze fra le operazioni, evitandole dove necessario.

Parte C

Progettare un servitore multiservizio (usando obbligatoriamente una select) che consenta di effettuare le operazioni remote per:

- *visualizzare tutte le calze di un certo tipo e carbone*;
- *inserire una nuova calza.*

Più in dettaglio:

- Il **client_stream** è organizzato come un **processo filtro (ciclico fino alla fine dell'input)** e realizza la funzionalità di **visualizzazione di tutte le calze di un certo tipo e carbone** utilizzando **una unica connessione** e **un'unica socket stream**.
  Per ogni richiesta, il client richiede all'utente il tipo di calza e l'indicazione di carbone e li invia al server, quindi riceve l'elenco di tutte le calze di quel tipo e che risultano con o senza carbone (per cui la struttura riporta 'S' o 'N') e li stampa a video.
- Il **client_datagram** è organizzato come un **processo filtro ciclico fino alla fine del file di input** e realizza la funzionalità di **inserimento di una nuova calza** utilizzando **socket datagram**.
  Per ogni richiesta, il client richiede all'utente e invia al server l'identificatore del bambino, il tipo, il carbone, la città, la via e il messaggio; quindi riceve l'esito dell'operazione e lo stampa a video.
- Il **server** principale unico discrimina le richieste utilizzando la primitiva select: il **server gestisce in modo parallelo** la funzionalità di visualizzazione di tutte le calze di un certo tipo e carbone; mentre la funzionalità di inserimento di una nuova calza **è realizzata in modo seriale.**
  Per ogni richiesta di **visualizzazione di tutte le calze di un certo tipo e carbone**, il figlio riceve il tipo di calza e l'indicazione di carbone e trasmette al client gli identificatori delle calze di quel tipo che risultano con/senza carbone.
  Per ogni richiesta di **inserimento di una nuova calza**, il server riceve l'identificatore del bambino, il tipo, il carbone, la città, la via e il messaggio e restituisce al client l'esito dell'operazione: valore positivo se l'inserimento ha successo, oppure un valore negativo, ad esempio se una calza con lo stesso identificatore è già stata inserita, se la struttura dati è piena o se falliscono alcuni controlli sui dati di input.

Parte Java

Utilizzando java RMI sviluppare un'applicazione C/S che consenta di effettuare le operazioni remote per:

- *visualizzare tutte le città dove saranno consegnate calze;*
- *visualizzare il numero di calze consegnate in una città e una via.*

Il progetto RMI si basa su un'interfaccia (contenuta nel file *RMI_interfaceFile.java*) in cui vengono definiti i metodi invocabili in remoto dal client:

- Il metodo **`visualizza_lista`** non ha parametri d'ingresso; il server restituisce la lista delle città dove saranno consegnate le calze (ovviamente evitando ripetizioni).
- Il metodo **`visualizza_numero`** accetta come parametri d'ingresso una città e una via; quindi il server restituisce l'esito dell'operazione, un intero che rappresenta il numero di calze da consegnare in quella città e via, -1 altrimenti, ad esempio, se città e via non sono presenti o uno dei controlli sugli input fallisce.

Si progettino le classi:

- ***RMI_Server*** (contenuta nel file *RMI_Server.java*), che implementa i metodi del server invocabili in remoto;
- ***RMI_Client*** (contenuta nel file *RMI_Client.java*), il processo filtro che realizza l'interazione con l'utente **proponendo ciclicamente i servizi** che utilizzano i due metodi remoti, e stampa a video i risultati, fino alla fine del file di input da tastiera.