Reti di Calcolatori T
Reti di Calcolatori L-A
Appello del 04/02/2011
Compito 3

Cognome: ……
Nome: ……
Matricola: ……

Tempo a disposizione: 2h

È obbligatorio inserire Cognome Nome Matricola e Numero Compito 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 sui parametri della richiesta e si gestiscano le eccezioni, 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 dei servizi **VotazioniPulite**, per la gestione delle votazioni alle elezioni politiche.
I servizi di VotazioniPulite utilizzano due strutture dati. La prima struttura dati, **Liste**, mantiene i nomi delle liste dei partiti ammessi alle elezioni e delle relative coalizioni; in particolare, per ogni lista si riportano: **il nome della lista**, unico all'interno del sistema; e **il nome della coalizione** di appartenenza, intesa come nome della coalizione a cui possono afferire più liste. La seconda struttura dati, **Candidati**, invece, mantiene i candidati; per ogni candidato riporta le seguenti informazioni: il **nome** del candidato, unico all'interno del sistema; la **lista di appartenenza** del candidato (una fra quelle presenti nella prima struttura dati), intesa come nome del partito a cui possono afferire più candidati; e il **numero di voti**, un intero inteso come numero di voti ottenuti finora dal candidato.
Si vogliono realizzare le funzionalità di gestione:

1. **inserimento di un nuovo candidato**: questa operazione richiede il nome del candidato e la lista di appartenenza, controlla l'esistenza della lista nella struttura dati Liste, e aggiunge il candidato impostando a 0 il numero di voti;
2. **visualizzazione del totale dei voti ottenuti da una coalizione**: questa operazione richiede il nome della coalizione e restituisce il totale dei voti ottenuti da tutti i candidati che appartengono a tutte le liste afferenti a tale coalizione;
3. **esprimere preferenza di un candidato**: questa operazione richiede il nome del candidato e aggiunge un voto; se la persona votata non esiste si conta un voto nullo.
4. **visualizzazione dei candidati di una lista**: questa operazione richiede il nome della lista e restituisce l'elenco di tutti i candidati appartenenti a tale lista.

Si progettino con particolare attenzione **le due strutture dati (Liste e Candidati)** che mantengono lo stato di liste e candidati, fino ad un massimo di N elementi ciascuna. In particolare, nella struttura dati Liste, esiste una lista fittizia, ad esempio chiamata "nulla" e con nome di coalizione "nulla", per contare i voti nulli. Le strutture dati sono da implementare opportunamente nei diversi ambienti richiesti, Java e C.
Si considerino e si segnalino le possibilità di interferenze fra le operazioni, evitandole dove necessario.

Parte C

Utilizzando RPC sviluppare un'applicazione C/S che consenta di effettuare le operazioni remote per:
- visualizzare il totale dei voti ottenuti da una coalizione;
- inserire un nuovo candidato.

Il progetto RPC si basa su un'interfaccia (contenuta nel file *RPC_xFile.x*) in cui vengono definiti i metodi invocabili in remoto dal client:
- La procedura **`visualizzazione_voti_lista`** accetta come parametro d'ingresso una stringa col nome della lista; quindi il server percorre la struttura dati valutando il totale dei voti ottenuti dai candidati afferenti alla coalizione richiesta (cioè a tutte le liste afferenti alla coalizione) e restituisce l'esito dell'operazione, un intero positivo pari ai voti totali se la valutazione è andata a buon fine, -1 altrimenti, ad esempio, se la coalizione cercata non è fra quelle ammesse.
- La procedura **`inserimento_candidato`** accetta come parametro d'ingresso la struttura dati NuovoCandidato contenente il nome del candidato e la lista d'appartenenza; quindi restituisce l'esito dell'operazione, 0 se l'inserimento è andato a buon fine, -1 altrimenti, ad esempio, se la lista non è fra quelle ammesse, se il nome è già presente, o se la struttura dati è piena.

Si progettino inoltre i programmi:
- ***RPC_Server*** (contenuta nel file *RPC_Server.c*), che implementa i metodi del server invocabili in remoto;
- ***RPC_Client*** (contenuta nel file *RPC_Client.c*), 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.

Parte Java

Usando socket stream e un'unica connessione per ogni sessione cliente, sviluppare un'applicazione C/S che consenta le operazioni remote per:
- votare per un candidato;
- visualizzare i candidati di una lista.

Più in dettaglio:
- Il **client** è organizzato come un processo ciclico fino alla fine del file di input. Per ogni iterazione del ciclo chiede all'utente quale tipo di operazione vuole effettuare e realizza le interazioni col server utilizzando **una sola connessione**. Alla ricezione del fine file, libera opportunamente le risorse e termina. Per ogni richiesta ricevuta dall'utente, il client prima invia il tipo di servizio al server, poi gestisce gli invii e le ricezioni necessarie alla realizzazione dello specifico servizio.
  Nel caso di **votazione di un candidato**, il client richiede all'utente e invia al server il nome del candidato, quindi riceve l'esito della votazione e lo stampa a video.
  Nel caso di **visualizzazione dei candidati di una lista**, il client richiede all'utente e invia al server il nome della lista; quindi riceve l'elenco dei candidati e lo stampa a video.

- Il **server** è organizzato come un demone e gestisce in modo parallelo l'interazione col client generando, all'arrivo di una sessione di richieste da un nuovo client, un figlio. Il figlio, con una prima lettura discrimina il tipo di funzionalità richiesto, poi gestisce opportunamente invii e ricezioni per l'operazione e si pone in attesa di nuove richieste dallo stesso client; alla lettura dell'EOF il figlio termina.
  Per ogni richiesta di **votazione di un candidato**, il figlio riceve il nome del candidato e controlla l'esistenza del candidato. Se il candidato esiste, incrementa il numero di voti del candidato e spedisce al client un messaggio di votazione riuscita, altrimenti incrementa il numero di voti nulli e spedisce al client un messaggio di votazione nulla.
  Per ogni richiesta di **visualizzazione dei candidati di una lista**, il figlio riceve il nome della lista e restituisce l'elenco di tutti i candidati che appartengono alla lista richiesta.