Reti di Calcolatori T
Appello del 12/09/2024
Compito 1

Cognome: ……

Nome: ……

Matricola: ……

Tempo a disposizione: 3h

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 **singolarmente tutti i file sorgente e tutti gli eseguibili prodotti** (*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.
In entrambi gli esercizi, sia in Java che in C, si effettuino gli opportuni controlli sugli argomenti della richiesta e si gestiscano le interazioni e 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 la progettazione di un servizio dedicato per la gestione dedicata a un appello di **RetiT**. Obbiettivo è l'insieme delle funzionalità tipiche rivolte a docenti – in generale, più di uno per corso - che permettono l'iscrizione, cancellazione e consultazione dello stato **lista esame**, considerando tutti i vincoli del caso (ad es. numero massimo d'iscrizioni). In particolare, si richiede di realizzare le seguenti funzionalità relative:

1. **iscrivi uno studente all'appello:** questa operazione richiede all'utente *la matricola, nome e cognome* e aggiunge lo studente al primo posto libero nella lista esame, restituendo a console l'esito dell'operazione.
2. **elimina una prenotazione dall'appello:** questa operazione richiede all'utente la ***matricola*** (ovvero *l'identificativo univoco studente*), eliminando la sua prenotazione e aggiornando la struttura dati, segnalando l'esito dell'operazione.
3. **visualizza lista studenti che hanno ottenuto un voto superiore a un valore soglia**: questa operazione richiede all'utente un valore *numerico* e restituisce con successiva stampa a video la lista degli studenti che hanno ottenuto un voto strettamente maggiore al valore soglia.
4. **carica voti**: questa operazione richiede all'utente un *insieme di coppie <matricola, voto>* e procede all'inserimento voti per ogni studente iscritto, segnalando eventuali errori in inserimento. L'errore tipico da considerare è *una mancata corrispondenza matricola nella lista iscritti, informando l'utente di tutte le occorrenze di errore*.

Si progetti con particolare attenzione la **struttura dati** (una **matrice**) **che mantiene lo stato e inizializzandola nel modo migliore**, fino ad un massimo di **N prenotazioni** (usando un modo per definire libero a default o mancata informazione), da implementare *opportunamente nei diversi ambienti richiesti, Java e C*.

max. 50

MATRICOLA	NOME	COGNOME	Voto
00012345	Pippo	Pluto	28
10101010	Giovanni	Verga	30 L
10101010	Giovanna	Pannone	-1
-	-	-	-
-	-	-	-
-	-	-	-

Parte C

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

- **eliminazione di una prenotazione**,
- **visualizza lista degli studenti che hanno ottenuto un voto superiore a un valore soglia**.

Il progetto prevede che il server metta a disposizione due procedure (parte di una interfaccia specificata in XDR e contenuta nel file RPC_xFile.x) invocabili in remoto dal client:

- La procedura **elimina_prenotazione** accetta come parametro in ingresso **una matricola** e procede a eliminare la prenotazione ad essa associata, restituendo **un intero positivo** in caso di successo, **oppure -1** in caso di *insuccesso, ad esempio in caso di mancata corrispondenza oaltro*.
- La procedura **vizualizza_voto_maggiore_soglia** accetta come parametro in ingresso **un numero** e restituisce una lista (array) di studenti che hanno ottenuto un voto strettamente maggiore al parametro in ingresso. Il metodo restituisce al più 5 risultati.

Si curi in particolare la struttura dati e la sua inizializzazione.

Si progettino inoltre i sorgenti:

- ***RPC_Server*** (contenuta nel file *RPC_Server.c*), che implementa le procedure del server invocabili in remoto;
- ***RPC_Client*** (contenuta nel file *RPC_Client.c*), il processo filtro che realizza l'interazione con l'utente, **propone ciclicamente i servizi** che utilizzano le due procedure remote, e stampa a video i risultati, fino alla fine dello stream di input.

Parte Java

Sviluppare un'applicazione C/S basata su **socket stream** che realizzi le operazioni remote per:

- **iscrivi studente all'appello;**
- **carica dei voti per un appello**,

utilizzando un'**unica connessione per ogni sessione cliente**.

- Il **cliente** è organizzato come un **processo filtro ciclico che consuma** l'input fino a fine **file** e, per ogni iterazione del ciclo, chiede all'utente quale tipo di operazione vuole effettuare e realizza le interazioni col server utilizzando **una sola connessione per la intera sessione**; alla ricezione di 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 richiesto. Nel caso di **iscrivi_studente_appello**, il client chiede all'utente e invia al server la *matricola*, nome e cognome dello studente, riceve l'esito dell'operazione e lo stampa a video. Nel caso della funzionalità di **carica_voto**, il client chiede all'utente un insieme di coppie *<matricola, voto>,* che invia al server e riceve l'esito dell'operazione stampando a video il risultato, in particolare di ogni coppia errata.
- Il **server** è organizzato come un **processo unico che gestisce in parallelo** l'interazione coi clienti, generando **un figlio per tutta la sessione di richieste di ogni cliente**. Per ogni richiesta, il figlio della sessione discrimina con una prima lettura il tipo di servizio richiesto, e gestisce opportunamente l'operazione, e dopo avere risposto, si pone in attesa di nuove richieste dal client; alla lettura di fine sessione, il processo figlio termina.
  Per ogni richiesta di **iscrivi_studente_appello**, il figlio legge *la matricola, nome e cognome* e procede a scandire la struttura dati interna trovando la prima posizione libera dove inserire la prenotazione, inviando come risposta al client l'esito dell'operazione ovvero 0 in caso *di successo*, *-1 in caso di impossibilità inserimento prenotazione*.
  Per ogni richiesta di **carica_voti_appello,** il figlio legge un insieme di coppie *<matricola, voto>* e per ogni matricola ricevuta procede all'inserimento voto, inviando come risposta al client l'esito dell'operazione ovvero, una lista vuota in caso di *successo* totale dell'operazione, altrimenti inviando una *lista di matricole non iscritte regolarmente all'esame*.