# Reti di Calcolatori T
# Appello del 13/01/2021
## Compito 2

**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 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 la progettazione e la realizzazione di **servizi invocabili** da parte di più clienti **sul file system di una macchina server** (direttorio remoto). I file binari possono avere qualsiasi estensione (non '.txt') e sono composti da dati binari. I file di testo sono i file con nome con suffisso '.txt' e composti solo da linee di caratteri ASCII.
In particolare, si realizzino i seguenti servizi:

1. **Conteggio delle linee che iniziano con una lettera minuscola e contengono almeno uno specificato numero di occorrenze di un carattere indicato all'interno di tutti i file di testo presenti sul direttorio remoto:** questa operazione richiede all'utente *un carattere e il numero di occorrenze*, quindi scandisce ciascun file di testo riga per riga contando il numero di righe che iniziano con una lettera minuscola e contengono (in qualsiasi posizione della riga) un numero di occorrenze del carattere uguale o superiore a quelle indicate dall'utente; infine, restituisce a console l'esito dell'operazione.
2. **Eliminazione di tutte le occorrenze di caratteri numerici all'interno di un file di testo:** questa operazione richiede all'utente il *nome di un file*, elimina tutte le occorrenze di *caratteri numerici* (da '0' a '9') dal file stesso, e visualizza sul cliente a video il numero di eliminazioni effettuate dal server o un'indicazione di errore.
3. **Lista dei sottodirettori di primo livello di un direttorio specificato che contengono almeno 6 file di testo**: questa operazione richiede all'utente un *nome di direttorio*, quindi visualizza la lista dei sottodirettori di primo livello del direttorio specificato che contengono almeno 6 file di testo (con suffisso '.txt').
4. **Trasferimento dal server al client di tutti i file di testo di un direttorio**: questa operazione richiede all'utente il nome del direttorio e trasferisce tutti i file testo in esso contenuti (i cui nomi abbiano suffisso '.txt'), salvandone il contenuto sul direttorio del client.

Ogni comando considera come direttorio di partenza il *direttorio corrente del client e del server*. Se per esempio il client richiede la lista dei sottodirettori (usando una notazione relativa) nel direttorio corrente, il server deve listare i sottodirettori del direttorio a partire dal corrente da cui è stato lanciato.
Si richiede inoltre di **non** usare nella soluzione comandi Unix (es. il comando **ls**), ma solo primitive di sistema (es. **opendir**() in C) e funzioni di libreria (es. metodi della **classe File** in Java).

## Parte Java

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

- eliminare **tutte le occorrenze di caratteri numerici** all'interno di un file di testo,
- ricevere **la lista dei sottodirettori di primo livello di un direttorio specificato che contengono almeno 6 file di testo**.

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 **elimina_occorrenze** accetta come parametro d'ingresso **il nome del file**, elimina tutte le occorrenze di caratteri numerici (da '0' a '9') all'interno di quello stesso file, e restituisce **un intero positivo** con il numero di eliminazioni effettuate (>=0) in caso di successo, **oppure -1** in caso di insuccesso, ad esempio errori o altro.
- Il metodo **lista_sottodirettori** accetta come parametro d'ingresso **il nome del direttorio**, e restituisce la lista **dei primi N** (**N<=6**) **nomi dei sottodirettori di primo livello trovati, se ci sono,** che contengono almeno 6 file di testo. In caso di direttorio inesistente, prevedere una segnalazione di errore.

Si progettino inoltre 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 che realizza l'interazione con l'utente, **propone 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 C

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

- **contare il numero delle linee che iniziano con una lettera minuscola e contengono almeno uno specificato numero di occorrenze di un carattere indicato** all'interno **di tutti i file di testo** presenti sul direttorio remoto,
- **trasferire** dal server al client **tutti i file di testo di un direttorio**,

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

Più in dettaglio:

- Il **client_stream** è organizzato come un **processo filtro ciclico fino alla fine del file di input** e realizza la funzionalità di **trasferimento dal server al client di tutti i file di testo di un direttorio** utilizzando **socket stream e un'unica socket**.
  Per ogni richiesta, il client per ogni richiesta, il client chiede all'utente il nome del direttorio, quindi invia al server il nome del direttorio e riceve i file salvandone il contenuto sul direttorio del client.
- Il **client_datagram** è organizzato come un **processo filtro ciclico fino alla fine del file di input** e realizza la funzionalità **conteggio del numero di linee** all'interno **di tutti i file di testo** presenti sul direttorio (corrente) remoto.
  Per ogni richiesta, il client chiede all'utente e invia al server il carattere e il numero di occorrenze, quindi riceve l'esito e lo stampa a video.
- Il **server** multiservizio discrimina le richieste utilizzando la primitiva select: il **server gestisce in modo parallelo** la funzionalità di trasferimento dal server al client di tutti i file di testo di un direttorio; mentre la funzionalità di conteggio del numero di linee con occorrenze carattere all'interno di tutti i file di testo presenti sul direttorio (corrente) remoto **è realizzata in modo seriale**.
  Per ogni richiesta di **trasferimento dal server al client di tutti i file di testo di un direttorio**, il figlio legge il nome del direttorio, quindi attua un protocollo per inviare al client tutti i file testo trovati.
  Per ogni richiesta di **conteggio del numero di linee** all'interno **di tutti i file di testo** presenti sul direttorio (corrente) remoto, il server legge il carattere e il numero di occorrenze, quindi esegue l'operazione di conteggio, scandendo ciascun file di testo riga per riga e contando il numero di righe che iniziano con una lettera minuscola e contengono (in qualsiasi posizione della riga) almeno un numero di occorrenze del carattere pari (cioè maggiore o uguale) a quelle indicate dall'utente, su ciascun file di testo trovato e invia la risposta al client, rappresentata da un intero positivo che indica il numero totale di occorrenze contate (>=0) in caso di successo, -1 in caso di problemi.