Homework Inverno 2019

Data di consegna: lunedì 6 gennaio, ore 24:00

DESCRIZIONE DELL'ASSEGNAMENTO

L'aeroporto locale della vostra città ha recentemente stretto un accordo con la famosa compagnia aerea low-cost AirNayr. L'incremento nei voli giornalieri ha portato a un aumento nel numero di passeggeri che purtroppo lamentano la carenza di un servizio parcheggi gestito in modo efficiente.
Per questo motivo il direttore dell'aeroporto vi ha assunto per progettare il sistema software che gestirà i due parcheggi che sta costruendo nelle vicinanze dell'aeroporto.
In una fase preliminare sono già state definite le principali specifiche che verranno descritte nel seguito. In fase di sviluppo potrà essere necessario procedere a un'ulteriore fase di analisi prendendo contatti con i responsabili del servizio per assicurarsi che lo strumento finale risponda alle loro esigenze.

SPECIFICHE

Nel seguito trovate una serie di specifiche che il vostro sistema dovrà rispettare. Si consiglia di procedere per punti e sviluppare il passo successivo solo quando il precedente funziona perfettamente.

CONTROLLO DEGLI ACCESSI

Una prima richiesta è di poter avere in ogni momento la lista delle auto presenti in ogni parcheggio. Ogni parcheggio è dotato di un solo ingresso e una sola uscita. I varchi di ogni parcheggio sono dotati di un sistema automatico per la lettura delle targhe. In ogni momento deve essere possibile accedere alla lista

delle targhe delle auto presenti in ogni parcheggio con l'informazione sull'ora di ingresso. All'uscita dal parcheggio la targa viene rimossa dalla lista. Se è stata raggiunta la capienza massima del parcheggio, l'auto non verrà fatta entrare.

RICHIESTE IMPLEMENTAZIONE

Per ogni parcheggio dovrete sviluppare il corpo di esecuzione di due thread. Una prima thread si occuperà di aggiornare i dati sulle auto in arrivo leggendo gli stessi da un file di input. Una seconda thread dovrà aggiornare le uscite, anche in questo caso leggendo i tempi di uscita da un file dato in input.
La scelta sulle strutture per memorizzare le informazioni sulle auto presenti nei parcheggi viene lasciata allo sviluppatore ma si raccomanda cautela sull'accesso alle risorse condivise.

DATI DI INPUT

Gli ingressi delle auto saranno memorizzati in due file: **`parcheggioUnoIngresso.txt`** e **`parcheggioDueIngresso.txt`.** Anche le informazioni sulle uscite sono memorizzate in file di testo **`parcheggioUnoUscita.txt`** e **`parcheggioDueUscita.txt`**.
Entrambi i file sono una sequenza di linee nel seguente formato
`TARGA_AUTO AA MM GG HH MM`
il tempo di ingresso (o di uscita) è in formato 24ore.
Si suppone che i file siano sempre in un formato corretto. Siete invitati a scambiarvi i file e confrontare le vostre soluzioni anche utilizzando il forum.

AGGIORNAMENTO SITUAZIONE CONTABILE

Una seconda richiesta del vostro committente riguarda la possibilità di mantenere la situazione contabile sempre aggiornata. In ogni momento dovrà essere possibile recuperare l'ammontare dei ricavi di ognuno dei parcheggi. Si considera che l'auto paga all'uscita del parcheggio.

RICHIESTE IMPLEMENTAZIONE

L'aggiornamento della situazione contabile dovrà essere svolto da un'ulteriore thread che si occupa di modificare il valore dei ricavi ad ogni uscita di auto da uno dei due parcheggi. La thread non dovrà contenere busy waiting (attese attive). Il costo orario di ogni parcheggio sarà memorizzato in due costanti che conterranno, rispettivamente, il costo giornaliero per ogni giorno intero passato nel parcheggio e il costo orario per ogni ora o frazione di ora.

PRENOTAZIONE PARCHEGGIO

Dopo le prime settimane di apertura ci si rende conto che è necessario realizzare anche un sistema di prenotazione via web. La durata di ogni prenotazione viene inserita nel sistema insieme alla targa associata. Un'auto prenotata troverà sempre il parcheggio disponibile, mentre un'auto non prenotata non potrà entrare se le auto già presenti più le auto con ingresso prenotato nella giornata raggiungono la capienza del parcheggio.

RICHIESTE IMPLEMENTAZIONE

Si supponga di ricevere delle richieste di prenotazioni da parte di clienti. Le prenotazioni vengono simulate da un'altra thread che leggerà da un file con il seguente formato:

```
AA MM GG HH MM TARGA AA MM GG AA MM GG
```

Dove il primo tempo `AA MM GG HH MM` indica il momento in cui avviene la prenotazione, seguito dalla targa e infine il giorno di arrivo e dal giorno di uscita della prenotazione. Si supponga che il file sia in formato corretto, i.e. non ci siano mai prenotazioni per giorni precedenti a quello di prenotazione.

ACCESSO SISTEMA DA REMOTO

L'ultima richiesta riguarda la possibilità di monitorare il sistema da remoto. Dovete realizzare un cliente collegato al sistema principale via Socket Stream che possa mandare i seguenti comandi:

1. Ricevere il ricavo attuale del parcheggio
2. Ricevere lo stato di entrambi i parcheggi (posti liberi, elenco targhe auto parcheggiate, elenco targhe auto prenotate).
3. Fermare il sistema: dovrete provvedere a fermare tutte le thread, lasciando il sistema in uno stato consistente.

RICHIESTE IMPLEMENTAZIONE

Si raccomanda di realizzare prima la comunicazione via Socket e, successivamente, realizzare un punto alla volta. Ognuno dei punti precedenti è a difficoltà crescente.

SUGGERIMENTI VARI

Le seguenti linee guida dovrebbero facilitare la consegna del vostro progetto. Ci aiuterete molto (e aiuterete la valutazione positiva della vostra prova) se osserverete i seguenti punti:

- Il programma dovrà essere scritto in modo modulare. Dopo aver terminato il vostro lavoro eseguite un git push sul repository.
- Il vostro assegnamento:
  - non deve contenere i file oggetto!
  - deve includere i file .c e .h
  - deve includere alcuni esempi di file di input
  - deve includere un CMakeLists.txt per compilare il vostro programma
  - deve includere un breve file README di spiegazione su quali test compiere (combinazione di file di input e argomenti a linea di comando)
- il progetto deve poter essere eseguito su piattaforme Linux
- utilizzate nomi ragionevoli per tutti i vostri file e le vostre variabili. In questo modo rendete più facile il nostro lavoro di correzione dei vostri progetti (e anche il vostro lavoro di debugging!!!)

- i vostri file non dovrebbero mai riferirsi a nomi assoluti. Per esempio per includere un file foo.h, non scrivete:

  #include ''/asf/...../foo.h''

- non utilizzate mai variabili globali (per chi non lo avesse capito abbiamo detto MAI) a meno che non siano dati condivisi tra le diverse thread

- siete incoraggiati a riutilizzare il vostro codice, che avete sviluppato in corsi precedenti, per code, sorting, liste,...

- siete anche incoraggiati ad utilizzare codice già fornito dalla C++ Standard Library

- Il progetto deve essere individuale. Non sarà tollerato l'uso di porzioni di codice prodotte da altri studenti o "consulenti". Chiedete consigli e chiarimenti, se necessario, solo dopo avere analizzato in profondità ed autonomamente il problema.

- se pensate che manchino delle specifiche al problema, per favore fate delle ipotesi ragionevoli e motivatele nel file di documentazione.

- se avete dei dubbi contattatemi (email: monica.reggiani@unipd.it, telefono: 393 5572915)... e non abbiate paura a chiamarmi...