ESAME DI PROGRAMMAZIONE C++ (8 CFU)

L'esame deve essere svolto singolarmente e deve essere realizzato unicamente con gli strumenti utilizzati nel corso. Dato che i progetti verranno testati con essi, ogni altro strumento potrebbe far fallire, ad esempio, la compilazione e quindi l'esame. In caso di esito negativo dell'esame, lo studente dovrà presentarsi ad un successivo appello d'esame con il progetto previsto per quella sessione.

Controllate spesso il sito del corso per eventuali aggiornamenti!

Questo documento contiene DUE progetti (leggere le note evidenziate):

Progetto C++

- Creazione di un programma a riga di comando con g++, make e doxygen
- Questo progetto deve essere svolto da tutti gli studenti.

Progetto Qt

- Creazione di un programma visuale con le librerie Qt
- Questo progetto deve essere svolto anche dagli studenti dell'insegnamento di "Programmazione e Amministrazione di Sistema" iscritti a partire dall'AA 17/18.
- **Gli studenti di Programmazione e Amministrazione di Sistema degli anni precedenti al 17/18 devono CONTATTARE IL DOCENTE.**

Progetto C++ del 27/06/2024

Data ultima di consegna: entro le 23.59 del 19/06/2024

Il progetto richiede la progettazione e realizzazione di una classe che implementa un **MultiSet ORDINATO** di elementi generici **T**. Un MultiSet è come un insieme di dati che può contenere duplicati: es. S={1,4,4,4,7,10,12}. Implementare il MultiSet in modo tale da minimizzare l'uso della memoria, cioè non dovete memorizzare i duplicati di un elemento. Facendo riferimento all'esempio precedente, il '4' va memorizzato una sola volta sapendo però che ci sono tre occorrenze di '4' in S.

A parte i metodi essenziali per la classe (tra cui conoscere il numero totale di elementi, aggiunta/rimozione elementi, conteggio occorrenze di un elemento, ecc...), devono essere implementate le seguenti funzionalità:

1. la costruzione di un **MultiSet** anche a partire da una sequenza di dati generici **Q** identificata da una coppia di iteratori generici. Questo costruttore prende in input: l'iteratore di inizio sequenza, l'iteratore di fine sequenza. Lasciate al compilatore la gestione della conversione di dati tra **Q** e **T**.

2. Un iteratore di sola lettura (scegliere la categoria). L'iteratore deve ritornare gli elementi del **MultiSet** tramite una struct pubblica **Pair** che contiene il valore e il numero delle sue occorrenze. Operativamente l'iteratore itera su una **Pair**.

3. Implementare l'operatore di confronto `operator==` tra due MultiSet che ritorna true sse i due **MultiSet** (dello stesso tipo) contengono gli stessi elementi con lo stesso numero di occorrenze dei duplicati.

4. Implementate un metodo `contains` che, dato un elemento di tipo T, ritorna true se l'elemento esiste nel **MultiSet**.

5. Implementare la funzione globale `operator<<` per inviare su std::ostream il contenuto di una **Pair** nella forma:
   <X, occorrenzeX>. Nell'esempio di S, per il valore 4 si avrà: <4, 3>

6. Implementare la funzione globale `operator<<` per inviare su std::ostream il contenuto del **MultiSet** nella forma:
   N <X1, occorrenzeX1>, <X2, occorrenzeX2>,...,<XN, occorrenzeXN>

Tenete anche conto che:

- La rimozione completa di un elemento X avviene quando il numero delle sue occorrenze diventa zero.
- Gli elementi del **MultiSet** e della **Pair** sono immutabili. Una volta inseriti, non cambiano valore.
- Non è necessario considerare questioni di efficienza del codice (a parte il requisito sulla memoria).

Utilizzare dove opportuno la gestione delle eccezioni.

Nota 1: Se non indicato diversamente, nella progettazione della classe, è vietato l'uso di librerie esterne e strutture dati container della std library come std::vector, std::list e simili. E' consentito il loro uso nel codice di test nel main.

Nota 2: A parte nullptr, non potete utilizzare altri costrutti C++11 e oltre se non indicato diversamente.

Nota 3: Nella classe, è consentito l'uso della gerarchia di eccezioni standard, delle asserzioni, la gerarchia degli stream e la funzione std::swap.

Nota 4: Per vostra sicurezza, tutti i metodi dell'interfaccia pubblica che implementate devono essere esplicitamente testati nel main anche su tipi custom. Evitate di fare dei test interattivi. Fato solo test automatici.

Nota 5: Non dimenticate di usare Valgrind per testare problemi di memoria

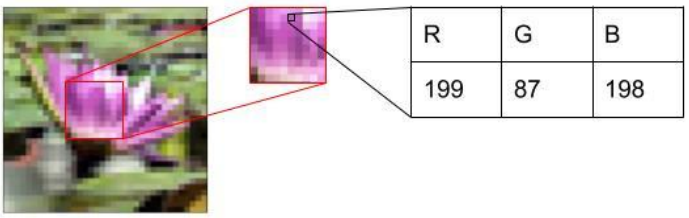
Nota 6: Evitate di usare "test" come nome dell'eseguibile. Potrebbe dare dei problemi sotto msys. Usate sempre il nome **main.exe**.

Alcune note sulla valutazione del Progetto C++

- Se in seguito a dei test effettuati dai docenti in fase di valutazione (es. chiamate a funzioni non testate da voi), il codice non compila, l’esame NON è superato.
- Implementazione di codice non richiesto non dà punti aggiuntivi ma se non corretto penalizza il voto finale.
- Gli errori riguardanti la gestione della memoria sono considerati GRAVI.
- La valutazione del progetto non dipende dalla quantità del codice scritto.
- NON usate funzionalità C di basso livello come memcpy, printf, FILE ecc... Se c'e' una alternativa C++ DOVETE usare quella.
- NON chiedete ai docenti se una VOSTRA scelta implementativa va bene o meno. Fa parte della valutazione del progetto.
- PRIMA DI SCRIVERE CODICE LEGGETE ACCURATAMENTE TUTTO IL TESTO DEL PROGETTO.

Progetto Qt del 27/06/2024

Data ultima di consegna: entro le 23.59 del 19/06/2024



Un'immagine RGB può essere interpretata come una matrice tridimensionale cui ciascun pixel contiene le intensità per i colori Rosso, Green e Blue (tripletta RGB). Nel seguente progetto è richiesto:

1. Poter caricare un'immagine RGB e visualizzarla.
2. Salvare le intensità delle triplette RGB dell'immagine usando la struttura dati della libreria Qt ritenuta idonea per:
    a. Non memorizzare la collocazione spaziale del pixel;
    b. Inserire le triplette secondo l'ordine scelto per la lettura dei pixel dell'immagine;
    c. Non memorizzare duplicati di una tripletta RGB. Per duplicati si intendono due triplette con le stesse intensità;
    d. Memorizzare una tripletta una sola volta insieme al numero di occorrenze nell'immagine.

es. **Pixel 127,145:** 199,87,198 **Pixel 32,14:** 32,17,80
→Non duplicati

199,87,198	1
32,17,80	1
…	…

Pixel 127,145: 199,87,198 **Pixel 132,145:** 199,87,198
→Duplicati

199,87,198	2
…	…

3. Visualizzare in un grafico il numero di occorrenze di ciascuna tripletta RGB normalizzato per il totale di pixel nell'immagine.
4. Salvare le intensità di ciascun canale R, G e B dell'immagine usando la struttura dati della libreria Qt ritenuta idonea per:
    a. Non memorizzare la collocazione spaziale del pixel;
    b. Memorizzate tutte le intensità ammissibili nel range ordinato da 0 a 255. Per le intensità non rappresentate nell'immagine il numero di occorrenze corrisponde a 0;
    c. Non memorizzare duplicati di una intensità;
    d. Memorizzare ciascuna intensità una sola volta insieme al numero di occorrenze nell'immagine.

    es. Struttura dati per le intensità del canale R:

0	13
1	154
2	12
…	…
255	0

5. Visualizzare in un grafico il numero di occorrenze per ciascuna intensità normalizzata per il totale di pixel nell'immagine.

Nota 1: Utilizzare preferibilmente la versione 5.12.11 della libreria Qt (la stessa installata sulla VM).

Nota 2: Si renda il contenuto dell'applicazione adattivo rispetto alla dimensione della finestra.

Alcune note sulla valutazione del Progetto Qt

- Se in seguito a dei test effettuati dai docenti in fase di valutazione (es. chiamate a funzioni non testate da voi), il codice non compila, l'esame NON è superato.
- Implementazione di codice non richiesto non dà punti aggiuntivi ma se non corretto penalizza il voto finale.
- NON verrà valutata l'efficienza dell'applicativo sviluppato.
- Gli errori riguardanti la gestione della memoria sono considerati GRAVI.
- La valutazione del progetto non dipende dalla quantità del codice scritto.
- NON chiedete ai docenti se una VOSTRA scelta implementativa o la configurazione dell'interfaccia grafica va bene o meno. Fà parte della valutazione del progetto.
- NON chiedete ai docenti come installare QtCreator e le librerie Qt

PRIMA DI SCRIVERE CODICE LEGGETE ACCURATAMENTE TUTTO IL TESTO DEL PROGETTO.

Consegna

La consegna del/dei progetti avviene tramite la piattaforma di eLearning ed è costituita da un archivio .tar.gz **avente come nome la matricola dello studente**. L'archivio deve contenere una e solo una cartella con lo stesso nome dell'archivio (senza estensione .tar.gz). Nella root della cartella devono essere presenti:

1. Un **makefile** (per poter compilare il progetto DA RIGA DI COMANDO) che deve compilare tutto il progetto chiamato "Makefile" (attenzione alle maiuscole). Se la compilazione fallisce, il progetto non viene considerato.
2. Tutti i **sorgenti** (commentati come avete visto ad esercitazione) del progetto e organizzati a vostro piacimento.
3. Il file di **configurazione di Doxygen** per la generazione della documentazione chiamato "Doxyfile" modificato per generare documentazione HTML.
4. **Relazione in PDF** con descrizione del progetto contenente informazioni relative al design e/o analisi del progetto. La relazione serve per capire il perchè delle vostre scelte nell'implementazione o di design. Nella relazione mettere anche Nome, Cognome, Matricola ed E-Mail.
5. **Chi deve consegnare anche il "Progetto Qt", metta tutti i file sorgenti corrispondenti in una sotto-cartella "Qt".**
6. L'archivio NON deve contenere file di codice oggetto, eseguibili etc..

L'eseguibile che verrà prodotto non deve richiedere alcun intervento esterno (es. input da tastiera).

Per creare l'archivio è sufficiente lanciare il comando (di msys o console Linux):

- tar –cvzf 123456.tar.gz 123456

dove "123456" è la directory che contiene tutti i file da consegnare.

Ad esempio, una struttura dell'archivio può essere questa:

```
123456
  |--main.cpp
  |--project.h
  |--Doxyfile
  |--…
  |--Qt  (SOLO PER PROGETTO Qt)
     |—*.pro
     |--MainWindow.cpp
     |--Main.cpp
     |--MainWindow.ui
     |--…
```