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 26/09/2022

Data ultima di consegna: entro le 23.59 del 17/09/2022

QUESTO PROGETTO E' VALEVOLE COME PROVA PARZIALE DEGLI INSEGNAMENTI DI "Programmazione ad Oggetti C++" (6CFU), "Programmazione C++" (4 CFU), "Programmazione e Amministrazione di Sistema" (8 CFU), "Programmazione C++" (8CFU)

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 tutti gli elementi del MultiSet. Cioè, i duplicati vanno ritornati in numero corretto. Nel caso d'esempio di S, è la sequenza di valori tipo: 1 4 4 4 7 10 12

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 del MultiSet nella forma: {<X1, occorrenzeX1>, <X2, occorrenzeX2>,...,<XN, occorrenzeXN>}

Tenete anche conto che:

- La rimozione completa di un elemento X avviene quando il numero di occorrenze di tale elemento diventa zero.
- Gli elementi del MultiSet 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.

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.

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.

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. Fà parte della valutazione del progetto.
- PRIMA DI SCRIVERE CODICE LEGGETE ACCURATAMENTE TUTTO IL TESTO DEL PROGETTO.

Progetto Qt del 26/09/2022

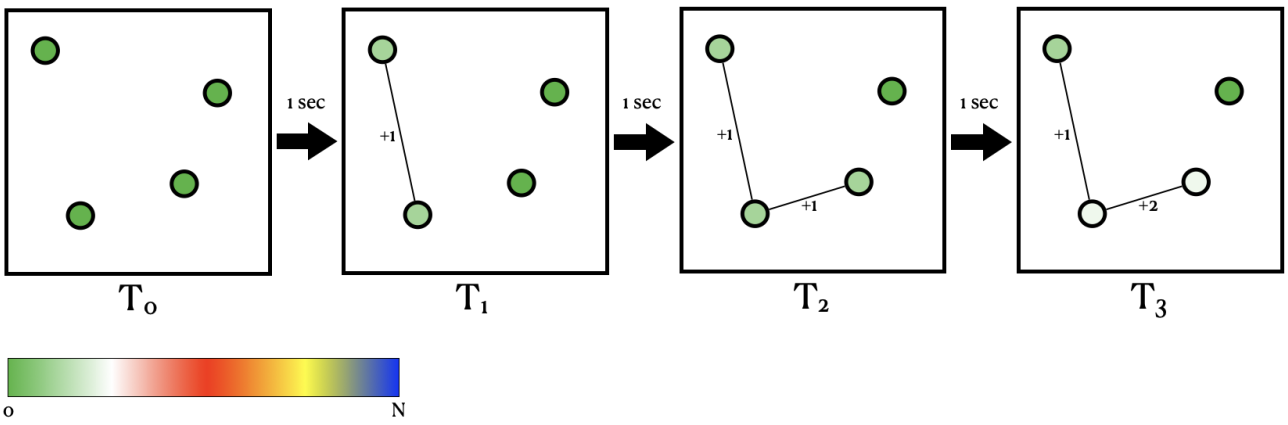
Data ultima di consegna: entro le 23.59 del 17/09/2022

QUESTO PROGETTO E' VALEVOLE COME PROVA PARZIALE DELL'INSEGNAMENTO DI "PROGRAMMAZIONE C++" (8CFU)
GLI STUDENTI ISCRITTI A PROGRAMMAZIONE E AMMINISTRAZIONE DI SISTEMA A PARTIRE DALL'AA 17/18 DEVONO SVOLGERE ANCHE QUESTO PROGETTO.

Si richiede la creazione di un'interfaccia grafica per la visualizzazione di un *dynamic interaction graph*. Quest'ultimo è un modello atto alla rappresentazione di interazioni sociali (ad esempio le interazioni sui social network). L'applicazione deve visualizzare un'istantanea del grafo delle interazioni.

- Lo stato del grafo viene aggiornato iterativamente con intervalli temporizzati di **un secondo**. L'applicazione deve visualizzare
- All'iterazione T_0, il grafo viene inizializzato con un numero casuale di nodi
- Alle iterazioni T_n si applicano le seguenti azioni:
  a. Si aggiunge un numero casuale di nuovi nodi (anche nessun nuovo nodo è contemplato);
  b. Si aggiunge un numero casuale di nuovi archi ponendo a +1 il **contatore delle interazioni** e modificando il colore dei due nodi cui l'arco è collegato (anche nessun nuovo arco è contemplato);
  c. Si incrementa di un'unità il contatore delle interazioni tra alcuni nodi del grafo scelti casualmente e si modifica coerentemente il colore dei due nodi che collegano l'arco.

Nella figura a seguire sono visualizzati diversi stati di avanzamento del grafo:



Altre importanti note:

- Nel caso di un nodo con più archi, il suo colore dipende dal valore più alto tra i contatori delle interazioni
- Il colore dei nodi deve essere codificato opportunamente utilizzando una colormap a scelta dello studente (come fonte d'ispirazione per la

creazione di una gradiente di colore si consulti il seguente link https://stackoverflow.com/questions/40132553/is-it-possible-to-change-the-color-of-a-qsliders-handle-according-to-its-positi)

- Per limitare il tempo di esecuzione e preservare la comprensibilità del grafo è opportuno limitare il numero massimo di nodi a 30.

Nota 1: Utilizzare la versione 5.12 o superiori della libreria Qt.
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 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. GMail ha bloccato l'invio di file con estensione .js quindi non è più possibile allegare la documentazione in formato 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:

- 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
     |--dataset
        |--panda
        |--saxophone
        |--…
     |--…
```

Indirizzo Mail: **corsocpp.ciocca@gmail.com**

Mettere come tag all'oggetto della mail:

[c++-consegna] Per consegnare ufficialmente il progetto
Potete fare più consegne e solo l'ultima verrà ritenuta valida.

~~[c++-test] Per testare il meccanismo di consegna~~
~~Verrà eseguita una compilazione del **Progetto C++** (differita e con cadenza oraria 0.00, 1.00, 2.00,...) e restituita una mail con l'esito. Potete fare tutti i test che volete. E' vivamente consigliato effettuare almeno una prova di compilazione (i progetti che non compilano, non vengono considerati). Il Progetto Qt non verrà compilato.~~

NOTA: questa mail deve essere usata esclusivamente per la consegna dei progetti. Per ogni altra questione usare le mail dei docenti.