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 C++ del 12/02/2020

Data ultima di consegna: entro le 23.59 del 31/01/2020

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 una matrice sparsa (**SparseMatrix**) di elementi generici **T**. Una matrice sparsa è una matrice nella quale solo gli elementi esplicitamente inseriti sono fisicamente memorizzati. Dal punto di vista logico la matrice sparsa si deve comportare come una normale matrice: alla richiesta dell'elemento in posizione (i,j) deve essere ritornato l'elemento inserito oppure un valore di default D. Questo valore di default è definito dall'utente e deve essere passato in fase di costruzione della matrice.

A parte i metodi essenziali per la classe (tra cui conoscere il numero di righe/colonne, numero degli elementi inseriti, il valore dell'elemento D, ecc...), devono essere implementate le seguenti funzionalità:

1. La matrice deve essere realizzata in modo tale che siano esplicitamente memorizzati solo gli elementi inseriti dall'utente (utilizzo di memoria minimale). In pratica se la matrice contiene logicamente NxM valori, non bisogna allocare NxM elementi **T** (e neanche una matrice di NxM puntatori a **T**).

2. La classe deve includere il supporto agli iteratori (lettura E scrittura) di tipo forward per l'accesso ai soli elementi inseriti nella matrice. L'iteratore deve ritornare una struct `element` che contiene solo dati e in particolare le coordinate (i,j) che non devono essere modificabili e il valore inserito in quella posizione. Gli elementi devono essere ritornati in base alla loro posizione logica nella matrice, da sinistra verso destra e dall'alto al basso (coordinate crescenti).

3. L'inserimento di un valore alle coordinate (i,j) deve avvenire con un metodo `add`.

4. La lettura di un valore alle coordinate (i,j) deve avvenire tramite `operator()`. Se la posizione non ha valore, deve essere ritornato il valore di default D.

5. Deve esistere anche un costruttore che costruisce la **SparseMatrix** definita su tipo **T**, da una **SparseMatrix** definita su tipo **Q**. Delegate al compilatore il controllo della convertibilità **Q**->**T**.

6. Scrivete una funzione globale generica `evaluate` che data una matrice sparsa M e un predicato P, ritorna quanti valori in M (compresi i default) soddisfano P. Testate diversi usi della funzione anche su diversi tipi di dati custom.

Utilizzare dove opportuno la gestione degli errori tramite asserzioni o 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: Non potete utilizzare i 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.