Fondamenti di Programmazione B

Prova di programmazione (C++)

Classe `Vettore<T>`

1) Realizzare in C++ una classe generica di nome `Vettore<T>` che implementa il tipo di dato astratto *vettore di elementi di tipo qualsiasi* `T`.

La classe fornisce le seguenti funzioni proprie *public*:
- funzione `operator[](i)`: restituisce l'elemento di indice `i` di questo vettore (solleva l'eccezione `"out of range"` se `i` è fuori dai limiti attuali del vettore);
- operatori = e ==: assegnamento e confronto tra questo vettore e un altro oggetto di tipo `Vettore<T>`;
- funzione `stampa(f_out)`: stampa tutti gli elementi di questo vettore sullo stream di output individuato da `f_out`, separati da uno spazio.

La classe fornisce anche:
- costruttore `Vettore(n)`: crea un vettore di `n` elementi di tipo `T`;
- costruttore `Vettore(n,v)`: crea un vettore di `n` elementi di tipo `T` e li inizializza tutti con il valore `v`;
- distruttore.

E' inoltre prevista una funzione che ridefinisce l'operatore << per la classe `Vettore<T>`.

Requisiti di implementazione. Realizzare il vettore tramite un array dinamico di elementi di tipo `T`. Non è prevista crescita dinamica dell'array. Utilizzare soltanto passaggio parametri *const reference* per oggetti di tipo `Vettore<T>`.

2) Scrivere un programma principale di prova che crea due oggetti `Vettore<T>`, uno di nome `A` per elementi di tipo `int` e uno di nome `B` per elementi di tipo `float`, entrambi di capacità massima *100* e con tutti gli elementi inizializzati a *-1*; quindi legge da std input un numero intero `i` ed assegna (tramite operatore `[]`) il numero `3` all'elemento di indice `i` di `A` e il numero `2.5` all'elemento di indice `i` di `B`; successivamente, stampa sia `A` che `B` su std output (tramite l'uso dell'operatore `<<`).
Infine crea un nuovo oggetto `Vettore<T>` per elementi di tipo `int` di nome `C`, assegna `A` a `C` e quindi controlla se `A` e `C` sono uguali, stampando il risultato del confronto su std output (`"vettori uguali"` o `"vettori diversi"`, rispettivamente).

Prevedere anche la gestione delle eventuali eccezioni generate dalla classe definita.