Esame 20240725

Esercizio 2

(1) **Esercizio 2 v1**

ESSAY | marked out of 10 | penalty 0 | File picker

Data una struttura dati `Queue` che rappresenta uno coda di interi (si veda il file `esercizio2.cpp`), e una serie di operazioni su questa struttura (e.g., `initQueue`, `isEmpty`, `enqueue`, `first`, `dequeue`, `length`, `reversedeleteQueue`, `printQueue`), si vuole implementare una nuova funzione ricorsiva `calcola` che prende come argomento due puntatori a `Queue q1` e `q2`. Le due code `q1` e `q2` hanno uguale lunghezza (si può quindi assumere che siano uguali senza doverlo verificare e/o gestire). Questa funzione **modifica la coda `q1`** in modo che ogni nodo in posizione `i` sia ottenuto sommando il valore in posizione `i` in `q1` con il contenuto nella posizione `N-i` nella coda `q2`, dove `N` è la lunghezza della coda `q1` (e quindi anche della coda `q2`).

La funzione `calcola`

- **deve essere ricorsiva** e **NON deve contenere iteratori** espliciti (`for`, `while`, `do-while`). La funzione `calcola` può ovviamente contenere codice sequenziale o condizionale. Sono consentite (se ritenute necessarie) chiamate a funzioni ricorsive ausiliarie che a loro volta **non contengano iterazioni esplicite** (`for`, `while`, `do-while`).
- **deve lasciare inalterato il contenuto** di `q2` alla fine dell'esecuzione (*ma lo può modificare se ritenuto necessario per la realizzazione*).
- **NON deve creare strutture intermedie** (e.g., array, code, stack, liste, ...) **dove memorizzare il contenuto della coda** `q1`. Valutare con attenzione le scelte implementative relative alle modalità di passaggio delle code `q1` e `q2` alla funzione `calcola`.
- **deve usare SOLO i metodi della coda** (i.e., `initQueue`, `isEmpty`, `enqueue`, `first`, `dequeue`, `length`, `reverse`,`deleteQueue`, `printQueue`) e **NON deve usare i dettagli implementativi della coda**, pena annullamento della prova.
- se ritenuto necessario è possibile definire funzioni ausiliarie ricorsive (e.g., `reverse`), che operano sulla coda, ma che usino solo i metodi della coda, e che non usino strutture intermedie (vedi punti precedenti).

Il file `esercizio2.cpp` contiene l'implementazione della struttura `Queue`, di alcuni metodi di utilità, e un `main` con alcuni esempi e alcune invocazioni della funzione `calcola`. Di seguito è riportato l'output di esecuzione.

```
marco > ./a.out
Q1: 10 8 6 4 2
Q2: 9 7 5 3 1
NQ1: 11 11 11 11 11
NQ2: 9 7 5 3 1
Q1: 7 7 8 6 0 8 6 8 6 0
Q2: 7 6 6 8 5 6 2 4 2 7
NQ1: 14 9 12 8 6 13 14 14 12 7
NQ2: 7 6 6 8 5 6 2 4 2 7
Q1: Queue is empty
Q2: Queue is empty
NQ1: Queue is empty
NQ2: Queue is empty
```

Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `calcola`, e **caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio** nello spazio apposito.
- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream`, `cstdlib`.
- Si ricorda che, gli esempi di esecuzione sono puramente indicativi, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.
- Si ricorda di inserire solo nuovo codice e di **NON MODIFICARE** il resto del programma (pena annullamento dell'esercizio).

esercizio2.cpp

Information for graders:

(2) **Esercizio 2 v2**

ESSAY | marked out of 10 | penalty 0 | File picker

Data una struttura dati `Queue` che rappresenta uno coda di interi (si veda il file `esercizio2.cpp`), e una serie di operazioni su questa struttura (e.g., `initQueue`, `isEmpty`, `enqueue`, `first`, `dequeue`, `length`, `reversedeleteQueue`, `printQueue`), si vuole implementare una nuova funzione ricorsiva `calcola` che prende come argomento due puntatori a `Queue` `q1` e `q2`. Le due code `q1` e `q2` hanno uguale lunghezza (si può quindi assumere che siano uguali senza doverlo verificare e/o gestire). Questa funzione **modifica la coda `q1`** in modo che ogni nodo in posizione `i` sia ottenuto moltiplicando il valore in posizione `i` in `q1` con il contenuto nella posizione `N-i` nella coda `q2`, dove `N` è la lunghezza della coda `q1` (e quindi anche della coda `q2`).

La funzione `calcola`

- **deve essere ricorsiva** e **NON deve contenere iteratori** espliciti (`for`, `while`, `do-while`). La funzione `calcola` può ovviamente contenere codice sequenziale o condizionale. Sono consentite (se ritenute necessarie) chiamate a funzioni ricorsive ausiliarie che a loro volta **non contengano iterazioni esplicite** (`for`, `while`, `do-while`).
- **deve lasciare inalterato il contenuto** di `q2` alla fine dell'esecuzione (*ma lo può modificare se ritenuto necessario per la realizzazione*).
- **NON deve creare strutture intermedie** (e.g., array, code, stack, liste, ...) **dove memorizzare il contenuto della coda** `q1`. Valutare con attenzione le scelte implementative relative alle modalità di passaggio delle code `q1` e `q2` alla funzione `calcola`.
- **deve usare SOLO i metodi della coda** (i.e., `initQueue`, `isEmpty`, `enqueue`, `first`, `dequeue`, `length`, `reverse`,`deleteQueue`, `printQueue`) e **NON deve usare i dettagli implementativi della coda**, pena annullamento della prova.
- se ritenuto necessario è possibile definire funzioni ausiliarie ricorsive (e.g., `reverse`), che operano sulla coda, ma che usino solo i metodi della coda, e che non usino strutture intermedie (vedi punti precedenti).

Il file `esercizio2.cpp` contiene l'implementazione della struttura `Queue`, di alcuni metodi di utilità, e un `main` con alcuni esempi e alcune invocazioni della funzione `calcola`. Di seguito è riportato l'output di esecuzione.

```
marco > ./a.out
Q1: 10 8 6 4 2
Q2: 9 7 5 3 1
NQ1: 10 24 30 28 18
NQ2: 9 7 5 3 1
Q1: 7 7 8 6 0 8 6 8 6 0
Q2: 7 6 6 8 5 6 2 4 2 7
NQ1: 49 14 32 12 0 40 48 48 36 0
NQ2: 7 6 6 8 5 6 2 4 2 7
Q1: Queue is empty
Q2: Queue is empty
NQ1: Queue is empty
NQ2: Queue is empty
```

Note:

- Scaricare il file `esercizio2.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `calcola`, e **caricare il file sorgente risultato delle vostre modifiche a soluzione di questo esercizio** nello spazio apposito.

- All'interno di questo programma **non è ammesso** l'utilizzo di variabili globali o di tipo `static` e di funzioni di libreria al di fuori di quelle definite in `iostream`, `cstdlib`.
- Si ricorda che, gli esempi di esecuzione sono puramente indicativi, e la soluzione proposta NON deve funzionare solo per l'input fornito, ma deve essere robusta a variazioni compatibili con la specifica riportata in questo testo.
- Si ricorda di inserire solo nuovo codice e di **NON MODIFICARE** il resto del programma (pena annullamento dell'esercizio).

esercizio2.cpp

Information for graders:

Total of marks: 20