

Esame 20250107

Esercizio 3

(1) Esercizio 3 v1

ESSAY marked out of 10 penalty 0 File picker

Data una struttura dati `Queue` che rappresenta una coda di interi (si veda il file `esercizio3.cpp`), e una serie di operazioni su questa struttura (e.g., `initQueue`, `isEmpty`, `enqueue`, `first`, `dequeue`, `length`, `reverse`, `deleteQueue`, `printQueue`), si vuole implementare una nuova funzione `calcola` che prende come argomento una `Queue q1`. Sia k il primo valore della coda, assumendo che non sia vuota. Questa funzione, se la coda non è vuota, **modifica la coda `q1`** in modo che i primi k elementi della coda siano invariati, e i restanti $N-k$ elementi siano rovesciati (ovvero in ordine inverso), dove N è la lunghezza della coda.

La funzione `calcola`:

- **NON deve creare strutture intermedie esplicite** (e.g., array, code, stack, liste, ...) **dove memorizzare il contenuto della coda `q1`**. Valutare con attenzione le scelte implementative relative alle modalità di passaggio della coda `q1` 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 in alcun modo i dettagli implementativi della coda**, pena annullamento della prova.
- se ritenuto necessario è possibile definire funzioni ausiliarie che operano sulla coda, ma che usino solo i metodi della coda, e che non usino strutture intermedie (vedi punti precedenti).
- deve gestire in modo corretto il caso in cui la coda sia vuota, il caso in cui la coda contenga un numero di elementi minore o uguale al valore di k , e il caso in cui la coda contenga un numero di elementi maggiore del valore di k .

Il file `esercizio3.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: 3 7 7 7 6 8 6 6 8 0 5
NQ1: 3 7 7 5 0 8 6 6 8 6 7
Q1: 3 8 6
NQ1: 3 8 6
Q1: 3 8 6
NQ1: 3 8 6
Q1: Queue is empty
NQ1: Queue is empty
```

Note:

- Scaricare il file `esercizio3.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).

`esercizio3.cpp`

Information for graders:

(2) Esercizio 3 v2

ESSAY marked out of 10 penalty 0 File picker

Data una struttura dati `Queue` che rappresenta una coda di interi (si veda il file `esercizio3.cpp`), e una serie di operazioni su questa struttura (e.g., `initQueue`, `isEmpty`, `enqueue`, `first`, `dequeue`, `length`, `reverse`, `deleteQueue`, `printQueue`), si vuole implementare una nuova funzione `calcola` che prende come argomento una `Queue q1`. Sia k il primo valore della coda, assumendo che non sia vuota. Questa funzione, se la coda non è vuota, **modifica la coda `q1`** in modo che i primi k elementi della coda siano rovesciati (ovvero in ordine inverso), e i restanti $N-k$ elementi siano invariati, dove N è la lunghezza della coda.

La funzione `calcola`:

- **NON deve creare strutture intermedie esplicite** (e.g., array, code, stack, liste, ...) **dove memorizzare il contenuto della coda `q1`**. Valutare con attenzione le scelte implementative relative alle modalità di passaggio della coda `q1` 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 in alcun modo i dettagli implementativi della coda**, pena annullamento della prova.
- se ritenuto necessario è possibile definire funzioni ausiliarie che operano sulla coda, ma che usino solo i metodi della coda, e che non usino strutture intermedie (vedi punti precedenti).
- deve gestire in modo corretto il caso in cui la coda sia vuota, il caso in cui la coda contenga un numero di elementi minore o uguale al valore di k , e il caso in cui la coda contenga un numero di elementi maggiore del valore di k .

Il file `esercizio3.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: 3 7 7 7 6 8 6 6 8 0 5
NQ1: 7 7 3 7 6 8 6 6 8 0 5
Q1: 3 8 6
NQ1: 6 8 3
Q1: 6 8 3
NQ1: 3 8 6
Q1: Queue is empty
NQ1: Queue is empty
```

Note:

- Scaricare il file `esercizio3.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).

esercizio3.cpp

Information for graders:

Total of marks: 20