Esame 20240620

Esercizio 3

(1) **Esercizio 3 v1**

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

Data una struttura dati `Stack` che rappresenta uno stack di interi (si veda il file `esercizio2.cpp`), e una serie di operazioni su questa struttura (e.g., `initStack`, `isEmpty`, `push`, `pop`, `top`, `printStack`, `deleteStack`, `lenght`), si vuole implementare una nuova funzione `calcola` che prende come argomento un puntatore a `Stack s`. Questa funzione **costruisci un nuovo stack** in modo che gli elementi dello stack `s` compaiono nello stesso ordine, e ogni elemento dello stack `s` è seguito immediatamente dal numero di occorrenze dello stesso valore che lo precedono (incluso il valore stesso) nello stack `s`. *Si assuma, per semplicità, che lo stack `s` contenga interi compresi tra 0 e 9.*

La funzione `calcola`

- **deve lasciare inalterato il contenuto** di `s` alla fine dell'esecuzione (*ma lo può modificare se ritenuto necessario per la realizzazione*).
- **NON deve creare strutture intermedie** (e.g., array, stack, liste, ...) **dove memorizzare il contenuto dello stack** `s`. Valutare con attenzione le scelte implementative relative alle modalità di passaggio dello stack `s` alla funzione `calcola`.
- **deve usare SOLO i metodi dello stack** (e.g., `initStack`, `isEmpty`, `push`, `pop`, `top`, `printStack`, `deleteStack`, `lenght`) e **NON deve usare i dettagli implementativi dello stack**, pena annullamento della prova.

Il file `esercizio3.cpp` contiene l'implementazione della struttura `Stack`, 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
Original before: 1 2 4 5
Result of calcola: 1 1 2 1 4 1 5 1
Original after: 1 2 4 5
Original before: 5 0 8 6 6 8 6 7 7 7
Result of calcola: 5 1 0 1 8 1 6 1 6 2 8 2 6 3 7 1 7 2 7 3
Original after: 5 0 8 6 6 8 6 7 7 7
Stack is empty
Stack 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 `Stack` che rappresenta uno stack di interi (si veda il file `esercizio2.cpp`), e una serie di operazioni su questa struttura (e.g., `initStack`, `isEmpty`, `push`, `pop`, `top`, `printStack`, `deleteStack`, `lenght`), si vuole implementare una nuova funzione `calcola` che prende come argomento un puntatore a `Stack s`. Questa funzione **costruisci un nuovo stack** in modo che gli elementi dello stack `s` compaiono nello stesso ordine, e ogni elemento dello stack `s` è seguito immediatamente dal numero di occorrenze dello stesso valore che lo seguono (incluso il valore stesso) nello stack `s`. *Si assuma, per semplicità, che lo stack `s` contenga interi compresi tra 0 e 9.*

La funzione `calcola`

- **deve lasciare inalterato il contenuto** di `s` alla fine dell'esecuzione (*ma lo può modificare se ritenuto necessario per la realizzazione*).
- **NON deve creare strutture intermedie** (e.g., array, stack, liste, ...) **dove memorizzare il contenuto dello stack** `s`. Valutare con attenzione le scelte implementative relative alle modalità di passaggio dello stack `s` alla funzione `calcola`.
- **deve usare SOLO i metodi dello stack** (e.g., `initStack`, `isEmpty`, `push`, `pop`, `top`, `printStack`, `deleteStack`, `lenght`) e **NON deve usare i dettagli implementativi dello stack**, pena annullamento della prova.

Il file `esercizio3.cpp` contiene l'implementazione della struttura `Stack`, 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
Original before: 1 2 4 5
Result of calcola: 1 1 2 1 4 1 5 1
Original after: 1 2 4 5
Original before: 5 0 8 6 6 8 6 7 7 7
Result of calcola: 5 1 0 1 8 2 6 3 6 2 8 1 6 1 7 3 7 2 7 1
Original after: 5 0 8 6 6 8 6 7 7 7
Stack is empty
Stack 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