Esame 20240620

Esercizio 2

(1) **Esercizio 2 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 ricorsiva `calcola` che prende come argomento due puntatori a `Stack` `s1` e `s2`. I due stack `s1` e `s2` hanno uguale lunghezza (si può quindi assumere che siano uguali senza doverlo verificare e/o gestire). Questa funzione **modifica lo stack `s1`** in modo che ogni nodo in posizione `i` sia ottenuto moltiplicando il valore in posizione `i` in `s1` con il contenuto nella posizione `N-i` nello stack `s2`, dove `N` è la lunghezza dello stack `s1` (e quindi anche `s2`).

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 `s2` 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** `s1`. Valutare con attenzione le scelte implementative relative alle modalità di passaggio dello stack `s1` 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 `esercizio2.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
S1: 2 4 6 8 10
S2: 1 3 5 7 9
NS1: 18 28 30 24 10
NS2: 1 3 5 7 9
S1: 0 6 8 6 8 0 6 8 7 7
S2: 7 2 4 2 6 5 8 6 6 7
NS1: 0 36 48 48 40 0 12 32 14 49
NS2: 7 2 4 2 6 5 8 6 6 7
Stack is empty
Stack is empty
Stack is empty
Stack 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 `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 ricorsiva `calcola` che prende come argomento due puntatori a `Stack` `s1` e `s2`. I due stack `s1` e `s2` hanno uguale lunghezza (si può quindi assumere che siano uguali senza doverlo verificare e/o gestire). Questa funzione **modifica lo stack `s1`** in modo che ogni nodo in posizione `i` sia ottenuto sommando il valore in posizione `i` in `s1` con il contenuto nella posizione `N-i` nello stack `s2`, dove `N` è la lunghezza dello stack `s1` (e quindi anche `s2`).

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 `s2` 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** `s1`. Valutare con attenzione le scelte implementative relative alle modalità di passaggio dello stack `s1` 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 `esercizio2.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
S1: 2 4 6 8 10
S2: 1 3 5 7 9
NS1: 11 11 11 11 11
NS2: 1 3 5 7 9
S1: 0 6 8 6 8 0 6 8 7 7
S2: 7 2 4 2 6 5 8 6 6 7
NS1: 7 12 14 14 13 6 8 12 9 14
NS2: 7 2 4 2 6 5 8 6 6 7
Stack is empty
Stack is empty
Stack is empty
Stack 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