Esame 20240904

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 `esercizio3.cpp`), e una serie di operazioni su questa struttura (e.g., `initStack`, `isEmpty`, `push`, `pop`, `top`, `printStack`, `deleteStack`, `lenght`), si vuole implementare un gioco, definendo una funzione `gioco` che prende come argomento due puntatori a `Stack s1` e `s2` e ritorna un intero. Le regole del gioco sono le seguenti:

- Ad ogni turno del gioco, si considera la somma dei valori al top di ogni stack modulo 10 (prima sommo e dopo faccio il modulo).
- Se tale somma è minore di 5, vince il turno lo stack `s1`, altrimenti vince il turno lo stack `s2`.
- Si rimuove un elemento dello stack perdente.
- Il gioco termina quando uno o entrambi gli stack sono vuoti, e lo stack non vuoto determina il vincitore.

La funzione `gioco` ritorna `1` se vince lo stack `s1`, ritorna `2` se vince lo stack `s2`, ritorna `0` se non vince nessuno (i.e., entrambi gli stack sono vuoti). La funzione `gioco` modifica gli stack `s1` e `s2` secondo le regole del gioco.

La funzione `gioco` **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` e dei metodi descritti sopra, e di alcuni metodi di utilità, e un `main` con alcuni esempi e alcune invocazioni della funzione `gioco`. Di seguito è riportato l'output di esecuzione.

```
marco > ./a.out
S1: 2 4 6 8 10
S2: 1 3 5 7 9
Stack is empty
NS2: 7 9
Lo stack vincente e': 2
S1: 28 12 0 19 16 10 16 8 26 28 10 26 18 17 27
S2: 24 22 37 16 38 37 2 24 12 36 35 28 6 26 37
NS1: 16 8 26 28 10 26 18 17 27
Stack is empty
Lo stack vincente e': 1
Stack is empty
Stack is empty
Stack is empty
Stack is empty
Lo stack vincente e': 0
```

Note:

- Scaricare il file `esercizio3.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `gioco`, 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 `esercizio3.cpp`), e una serie di operazioni su questa struttura (e.g., `initStack`, `isEmpty`, `push`, `pop`, `top`, `printStack`, `deleteStack`, `lenght`), si vuole implementare un gioco, definendo una funzione `gioco` che prende come argomento due puntatori a `Stack` `s1` e `s2` e ritorna un intero. Le regole del gioco sono le seguenti:

- Ad ogni turno del gioco, si considera la somma dei valori al top di ogni stack modulo 10 (prima sommo e dopo faccio il modulo).
- Se tale somma è minore di 5, vince il turno lo stack `s2`, altrimenti vince il turno lo stack `s1`.
- Si rimuove un elemento dello stack perdente.
- Il gioco termina quando uno o entrambi gli stack sono vuoti, e lo stack non vuoto determina il vincitore.

La funzione `gioco` ritorna `1` se vince lo stack `s1`, ritorna `2` se vince lo stack `s2`, ritorna `0` se non vince nessuno (i.e., entrambi gli stack sono vuoti). La funzione `gioco` modifica gli stack `s1` e `s2` secondo le regole del gioco.

La funzione `gioco` **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` e dei metodi descritti sopra, e di alcuni metodi di utilità, e un `main` con alcuni esempi e alcune invocazioni della funzione `gioco`. Di seguito è riportato l'output di esecuzione.

```
marco > ./a.out
S1: 2 4 6 8 10
S2: 1 3 5 7 9
NS1: 8 10
Stack is empty
Lo stack vincente e': 1
S1: 28 12 0 19 16 10 16 8 26 28 10 26 18 17 27
S2: 24 22 37 16 38 37 2 24 12 36 35 28 6 26 37
Stack is empty
NS2: 24 12 36 35 28 6 26 37
Lo stack vincente e': 2
Stack is empty
Stack is empty
Stack is empty
Stack is empty
Lo stack vincente e': 0
```

Note:

- Scaricare il file `esercizio3.cpp`, modificarlo per inserire la dichiarazione e la definizione della funzione `gioco`, 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