Esame 20250616

Esercizio 3

(1) **Esercizio 3 v1**

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

Nella trasmissione dati, i messaggi sono spesso decomposti in blocchi di dati, che vengono poi trasmessi in sequenza, e caratterizzati da un identificativo. I messaggi infatti arrivano a destinazione e memorizzati in uno stack nell'ordine corretto, ma in modo asincrono, quindi nello stack non si é garantiti che tutti i messaggi con lo stesso id siano uno di seguito all'altro. Quindi nello stack possono essere presenti blocchi di dati con lo stesso id, ma non necessariamente in diretta sequenza. Per poter processare i messaggi, é quindi necessario processare lo stack e prelevare i blocchi di dati con lo stesso id, in modo da poterli processare in sequenza, lasciando inalterati gli altri blocchi di dati.

Si assuma data una struttura `Message` (definita in `stack.h`) con due campi: `id` di tipo intero, e `messagge` di tipo array di `char` di dimensione `MAXSIZE+1` che é una stringa (quindi terminata con `'\0'`. Si assuma inoltre di avere una implementazione dello stack (definita in `stack.h` e `stack.cpp`) che permette di inserire ed estrarre i blocchi di dati dallo stack.

Si scriva una funzione `estrai_messaggio` che prende in input uno stack di `Message`, un intero che rappresenta l'identificatore del messaggio da estrarre, e un array di `char` che rappresenta la stringa del messaggio completo con lo stesso id (da allocare dinamicamente). Si valuti con attenzione la modalitá di passaggio dei tre parametri della funzione `estrai_messaggio`.

La funzione `estrai_messaggio` estrae (li rimuove) dallo stack i messaggi con lo stesso id, e concatena (*nell'ordine in cui compaiono*) i caratteri dei singoli messaggi nello stesso ordine in cui compaiono nello stack, e memorizza il risultato nel terzo parametro, che deve essere allocato dinamicamente. Nel caso non compaiano messaggi con lo stesso id, la funzione deve restituire una stringa vuota. I messaggi nello stack che non hanno l'identificativo cercato, rimangono nello stack nell'ordine in cui erano presenti prima della chiamata alla funzione `estrai_messaggio`. É giá fornita una funzione `mystrrevcat` che concatena due stringhe in modo inverso, e una funzione `mystrcat` che concatena due stringhe in modo diretto. Queste due funzioni, se ritenuto necessario, possono essere utilizzate per implementare la funzione `estrai_messaggio` se lo si ritiene necessario.

É giá fornita l'implementazione dello stack, che possono essere utilizzate per implementare la funzione `estrai_messaggio`. (Si vedano i file `stack.h`, `stack.cpp`).

Il file `esercizio3.cpp` contiene tutto quanto necessario tranne la dichiarazione e la definizione della funzione `estrai_messaggio`.

Di seguito sono riportati alcuni esempi di esecuzione del programma.

```
computer > ./a.out
Messaggio: 2
  Il professore ha molta fantasia!
Messaggio: 1
  Hello World!
Messaggio: 4

Stack --
3: Difficile!
3: molto
3: Esame
```

Note:

- Scaricare il file `esercizio3.cpp`, modificarlo per inserire il codice necessario per rispondere a questo esercizio. **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`, `fstream`, `cstring`.
- 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).
- Si ricorda che tutta la memoria allocata dinamicamente deve essere deallocata.

esercizio3.cpp

stack.cpp stack.h

Information for graders:

(2) **Esercizio 3 v2**

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

Nella trasmissione dati, i messaggi sono spesso decomposti in blocchi di dati, che vengono poi trasmessi in sequenza, e caratterizzati da un identificativo. I messaggi infatti arrivano a destinazione e memorizzati in uno stack nell'ordine corretto, ma in modo asincrono, quindi nello stack non si é garantiti che tutti i messaggi con lo stesso id siano uno di seguito all'altro. Quindi nello stack possono essere presenti blocchi di dati con lo stesso id, ma non necessariamente in diretta sequenza. Per poter processare i messaggi, é quindi necessario processare lo stack e prelevare i blocchi di dati con lo stesso id, in modo da poterli processare in sequenza, lasciando inalterati gli altri blocchi di dati.

Si assuma data una struttura `Message` (definita in `stack.h`) con due campi: `id` di tipo intero, e `messagge` di tipo array di `char` di dimensione `MAXSIZE+1` che é una stringa (quindi terminata con `'\0'`. Si assuma inoltre di avere una implementazione dello stack (definita in `stack.h` e `stack.cpp`) che permette di inserire ed estrarre i blocchi di dati dallo stack.

Si scriva una funzione `estrai_messaggio` che prende in input uno stack di `Message`, un intero che rappresenta l'identificatore del messaggio da estrarre, e un array di `char` che rappresenta la stringa del messaggio completo con lo stesso id (da allocare dinamicamente). Si valuti con attenzione la modalitá di passaggio dei tre parametri della funzione `estrai_messaggio`.

La funzione `estrai_messaggio` estrae (li rimuove) dallo stack i messaggi con lo stesso id, e concatena (*nell'ordine inverso a quello in cui compaiono*) i caratteri dei singoli messaggi nello stesso ordine in cui compaiono nello stack, e memorizza il risultato nel terzo parametro, che deve essere allocato dinamicamente. Nel caso non compaiano messaggi con lo stesso id, la funzione deve restituire una stringa vuota. I messaggi nello stack che non hanno l'identificativo cercato, rimangono nello stack nell'ordine in cui erano presenti prima della chiamata alla funzione `estrai_messaggio`. É giá fornita una funzione `mystrrevcat` che concatena due stringhe in modo inverso, e una funzione `mystrcat` che concatena due stringhe in modo diretto. Queste due funzioni, se ritenuto necessario, possono essere utilizzate per implementare la funzione `estrai_messaggio` se lo si ritiene necessario.

É giá fornita l'implementazione dello stack, che possono essere utilizzate per implementare la funzione `estrai_messaggio`. (Si vedano i file `stack.h`, `stack.cpp`).

Il file `esercizio3.cpp` contiene tutto quanto necessario tranne la dichiarazione e la definizione della funzione `estrai_messaggio`.

Di seguito sono riportati alcuni esempi di esecuzione del programma.

```
computer > ./a.out
Messaggio: 2
  !aisatnaf atlom ah erosseforp lI
Messaggio: 1
  !dlroW olleH
Messaggio: 4

Stack --
3: Difficile!
3: molto
3: Esame
```

Note:

- Scaricare il file `esercizio3.cpp`, modificarlo per inserire il codice necessario per rispondere a questo esercizio. **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`, `fstream`, `cstring`.
- 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).
- Si ricorda che tutta la memoria allocata dinamicamente deve essere deallocata.

esercizio3.cpp

stack.cpp stack.h

Information for graders:

Total of marks: 20