IP a.a. 2021/22 - Esame del 16 giugno 2022

Prima di cominciare lo svolgimento leggete attentamente tutto il testo.

Questa prova è organizzata in due sezioni, in cui sono dati alcuni elementi e voi dovete progettare ex novo tutto quello che manca per arrivare a soddisfare le richieste del testo. Per ciascuna sezione nel file zip del testo trovate una cartella contenente i file da cui dovete partire. Dovete lavorare solo sui file Cognome1.cpp e Cognome2.cpp, rinominandoli rispettivamente con il vostro cognome seguito dal numero 1 o 2. Modificare gli altri file è sbagliato (ovviamente a meno di errata corrige indicata dai docenti).

In questi file dovete realizzare le funzioni richieste, esattamente con la *segnatura* con cui sono indicate: nome, tipo restituito, tipo degli argomenti nell'ordine in cui sono dati. Potete inoltre realizzare altre funzioni in tutti i casi in cui lo ritenete appropriato. Potete inserirvi tutti gli `#include` che vi servono oltre a quello relativo allo header con le funzioni da implementare.

1 Istogramma

Un istogramma è una lista di valori misurati, a ciascuno dei quali è associato il numero di volte che il valore stesso è stato osservato. Tipicamente è mostrato graficamente come un **diagramma a barre**.

Vedere il file `hist.h` per le definizioni effettive.

Useremo una `struct` di tipo `Bin` per ogni valore, che assumiamo **intero** e **non negativo**. Ogni `Bin` contiene un membro di tipo `int` per memorizzare il valore e un altro membro, sempre di tipo `int`, per memorizzare il relativo numero di occorrenze.

Un istogramma sarà realizzato come un `std::vector` di elementi di tipo `Bin`, ridefinito con il nome `Hist`.

Materiale dato

Nel file zip trovate

- un file `hist.h` contenente le definizioni di tipo date e le intestazioni delle funzioni ⟵ NON MODIFICARE
- un file `histtest.cpp` contenente un main da usare per fare testing delle vostre implementazioni e la realizzazione (corpo) delle funzioni fornite da noi. ⟵ NON MODIFICARE
- un file `Cognome1.cpp` contenente lo scheletro delle funzioni che dovete realizzare voi ⟵ DOVETE MODIFICARE SOLO QUESTO, INCLUSO IL SUO NOME
- un file `valori.txt` contenente un testo di prova

Funzione `add` (DA FARE)

```
void add(Hist & h, int v);
```

Aggiunge il valore `v` al Hist `h`:

- se `v` non è presente, crea una nuova Bin con conteggio (`num`) pari a 1 e la aggiunge a h;
- se `v` è già presente, incrementa il suo conteggio.

Funzione `sort` (DA FARE)

```
void sort(Hist & h);
```

Ordina `h` secondo il valore contenuto (campo `value`).

Funzione `count` (DA FARE)

```
int count(const Hist & h, int v);
```

Restituisce il conteggio del valore `v` nel Hist `h`, oppure restitruisce 0 se `v` non è presente.

Funzione `print` (FORNITA)

```
void print(const Hist & h);
```

Stampa `h` su `std::cout`.

2 Puntatori: lista collegata semplice per scrittura inversa

Per questa parte lavorate nella cartella `Sezione2`, nel file `<VostroCognome>2.cpp`.

In questa sezione usiamo una lista collegata in cui aggiunta e rimozione di elementi avvengono sempre e solo dalla testa.

Il programma di test usa questa struttura dati per invertire l'ordine dei caratteri di una stringa.

La struttura dati è definita nel file header ll.h che dichiara, come d'abitudine, un tipo struct `elem` come tipo dell'elemento della lista, che ha come tipo base il singolo carattere, e un tipo puntatore a `elem`, chiamato `ll`.

L'obiettivo è creare le funzioni `addhead`, `removehead` e `size`.

Funzione `addhead` (PSEUDOCODICE FORNITO)

Scrivere una funzione che riceve in ingresso una lista collegata `s` (per riferimento) e un carattere `c`, e aggiunge in testa a `s` un nuovo elemento che contiene `c`.

Implementare la funzione seguendo la seguente procedura:

```
void addhead(ll & s, char & c)
{
    //  dichiarare una variabile tmp di tipo puntatore a elem
    //  allocare elem
    //  assegnare il valore di s all'elemento next di temp
    //  assegnare il valore di c all'elemento c di temp
    //  assegnare il valore di temp a s
}
```

Funzioni `removehead` e `size`

Scrivere due funzioni:

- La funzione `removehead` riceve in ingresso (per riferimento) la lista `s` e un carattere `c`. Se `s` non è vuota, assegna a `c` il valore contenuto nel membro `c` dell'elemento in testa a `s`, rimuove tale elemento, e restituisce `true`. Se `s` è vuoto, restituisce `false` senza fare altri cambiamenti.

  ```
  bool size(ll & s, char & c);
  ```

- La funzione `size` riceve in ingresso la lista `s` e ne restituisce il numero di elementi.

  ```
  int size(ll s);
  ```