IP a.a. 2023/24 - Esame Generato 2024

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 indicati in ciascuna parte. 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. Non è consentito modificare queste informazioni. Potete invece fare quello che volete all'interno del corpo delle funzioni: in particolare, se contengono già una istruzione return, questa è stata inserita provvisoriamente per rendere compilabili i file ancora vuoti, e **dovrete modificarla in modo appropriato**.

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. Attenzione però che **usare una funzione di libreria per evitare di scrivere del codice richiesto viene contato come errore** (esempio: se è richiesto di scrivere una funzione di ordinamento, usare la funzione std::sort() dal modulo di libreria standard algorithm è un errore).

Il programma principale, che esegue il test, è dato in ognuna delle sezioni. Controllate durante l'esecuzione del main, per ogni funzione, quanti sono i test che devono essere superati e controllate l'esito (se non ci sono errori deve essere true per tutti).

Compilare con: g++ -std=c++11 -Wall *.cpp

NB1: soluzioni particolarmente inefficienti potrebbero non ottenere la valutazione anche se forniscono i risultati attesi. Di contro ci riserviamo di premiare con un bonus soluzioni particolarmente ottimali.
NB2: superare positivamente tutti i test di una funzione non implica soluzione corretta e ottimale (e quindi valutazione massima).

1 Sezione 1 - Array - (max 7.5 punti)

Per questa parte lavorate nella cartella Sezione1. Per ogni esercizio dovete scrivere una funzione nel file specificato, fornito nel file zip, completandolo secondo le indicazioni.

Materiale dato

Nel file zip trovate

- Un file array.h contenente le definizioni di tipo dato e le intestazioni delle funzioni ←– NON MODIFICARE
- un file mainTestArray.cpp contenente un main da usare per fare testing ←– NON MODIFICARE
- un file Es1-Funzione1.cpp ←– MODIFICARE IL SUO CONTENUTO
- un file Es1-Funzione2.cpp ←– MODIFICARE IL SUO CONTENUTO
- un file Es1-Funzione3.cpp ←– MODIFICARE IL SUO CONTENUTO

1.1 Es1-Funzione1 - (2.5 punti)

```
Unsigned  int countPrimes( const int*arr, usnigned int size)
```

- INPUT:
  - const int*arr,: un array di interi,
  - usnigned int size: la dimensione dell'array
- OUTPUT: Il numero di numeri primi presenti nell'array
- Comportamento:
  La funzione deve contare quanti numeri primi sono presenti nell'array e restituire questo numero.
- Esempi:

INPUT => OUTPUT

Esempi:

arr=[] , size=0 => 0

arr=[2, 3, 4, 5, 6, 7] , size=6 => 4

arr=[1, 2, 3, 4, 5] , size=5 => 3

1.2 Es1-Funzione2 - (2 punti)

```
void rotateArray(int *arr, unsigned int size, int positions)
```

- INPUT:

  - int* arr: un array di interi

  - unsigned int size: la dimensione dell'array

  - int positions: il numero di posizioni da ruotare l'array

  Comportamento: La funzione deve ruotare l'array di positions posizioni a destra. Se positions è negativo, ruotare a sinistra.

- Esempi:
  - arr=[1, 2, 3], size=3, positions=1 => arr=[3, 1, 2]
  - arr=[1, 2, 3], size=3, positions=-1 => arr=[2, 3, 1]
  - arr=[1, 2, 3, 4], size=4, positions=2 => arr=[3, 4, 1, 2]

1.3 Es1-Funzione3 - (3 punti)

```
Int max maxSubArraySum( const int *arr, unsigned   Int size)
```

- INPUT:
  - const int *arr : un array di interi
  - unsigned int size: la dimensione dell'array
- OUTPUT: La somma massima di un sottoarray contiguo nell'array.
- Comportamento: La funzione deve trovare la somma massima di un sottoarray contiguo nell'array e restituire questa somma.
- Esempi: INPUT => OUTPUT

  - arr=[] , size=0 => 0

  - arr=[-2,1,-3,4,-1,2,1,-5,4] , size=9 => 6

  - arr=[1,2,3,4] , size=4 => 10

2 Sezione 2 - Liste - (max 8.5 punti)

Per questa parte lavorate nella cartella Sezione2. Per ogni esercizio dovete scrivere una funzione nel file specificato, fornito nel file zip, completandolo secondo le indicazioni.

Siano date le seguenti definizioni:

```
typedef std::string Elem;

struct Cell {
    Elem elem;
    struct Cell* next;
}; typedef Cell* List;
```

Si richiede di implementare le funzioni descritte nel seguito.

Materiale dato

Nel file zip trovate

- un file list.h contenente le definizioni di tipo dato e le intestazioni delle funzioni ←− NON MODIFICARE
- un file mainTestList.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 Es2-Funzione1.cpp ←− MODIFICARE IL SUO CONTENUTO
- un file Es2-Funzione2.cpp ←− MODIFICARE IL SUO CONTENUTO
- un file Es2-Funzione3.cpp ←− MODIFICARE IL SUO CONTENUTO

2.1 Es2-Funzione1 - (2 punti)

```
unsigned int computeListSize(const List &l)
```

- INPUT:
  - l: la lista della quale calcolare la lunghezza
- OUTPUT: la lunghezza della lista espressa come numero di elementi che la compongono.
- Comportamento:

  la funzione restituisce 0 se la lista l è vuota oppure un valore pari numero di elementi che la compongono. La funzione non deve modificare la lista l.

2.2 Es2-Funzione2 - (3 punti)

```
bool insertElemInListAtIndex(List &l, Elem e, unsigned int index )
```

- INPUT:
  - l: la lista nella quale inserire l'elemento
  - e: l'elemento da inserire nella lista
  - unsigned int index: la posizione alla quale inserire l'elemento
- OUTPUT: true se l'elemento è stato inserito correttamente, false altrimenti.Comportamento:

  la funzione deve inserire l'elemento s nella lista l alla posizione index.

2.3 Es2-Funzione3 - (3.5 punti)

```
void deleteLastInstanceOfElemInList (List &l, Elem e)
```

- INPUT:
  - l: la lista dalla quale eliminare gli elementi
  - e: l'elemento da eliminare

- Comportamento:la funzione deve eliminare l’ultima istanza dell’elemento s dalla lista l.