

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 correzione 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)

```
int sommaArray(int arrayInteri[], dimensione)
```

- INPUT:
 - `arrayInteri[]`: un array di interi,
 - `dimensione`: la dimensione dell'array
- OUTPUT: - La somma di tutti gli elementi dell'array.
- Comportamento:
la funzione prende in input un array di interi e la sua dimensione e restituisce la somma di tutti gli elementi presenti nell'array.
- Esempi:

INPUT => OUTPUT

-arrayinteri = [], dimensione = 0 => 0

-arrayinteri = [1,2,3], dimensione = 3 => 6

1.2 Es1-Funzione2 - (2 punti)

```
int trovaMax(int arrayInteri[], int dimensione)
```

- INPUT:
 - `arrayInteri[]`: un array di interi,
 - `dimensione`: la dimensione dell'array
- Comportamento: La funzione prende in input un array di interi e la sua dimensione e restituisce l'elemento massimo presente nell'array. Se l'array è vuoto, la funzione deve restituire INT_MIN.
- Esempi:
 - `arrayInteri=[], dimensione = 0 => INT_MIN`
 - `arrayInteri=[1,2,3], dimensione = 3 => 3`

1.3 Es1-Funzione3 - (3 punti)

```
std::vector<int> filtraPari(const int arrayInteri[], int dimensione)
```

- INPUT:
 - `arrayInteri[]`: un array di interi,
 - `dimensione`: la dimensione dell'array
- OUTPUT: *Un vettore contenente solo gli elementi pari dell'array di input.*
- Comportamento: La funzione prende in input un array di interi e la sua dimensione e restituisce un vettore contenente solo gli elementi pari presenti nell'array di input.
- Esempi: INPUT => OUTPUT
 - `arrayInteri=[1, 2, 3, 4], dimensione=4 => [2, 4]`
 - `arrayInteri=[1, 3, 5], dimensione=3 => []`

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 contaElement(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 aggiungiInTesta(List &l, Elem e, )
```

- INPUT:
 - `l`: la lista nella quale inserire l'elemento
 - `e`: l'elemento da inserire nella lista
- OUTPUT: Nessuno
- Comportamento:
la funzione deve inserire l'elemento in testa. La lista può contenere già valori oppure essere vuota (NULL)

2.3 Es2-Funzione3 - (3.5 punti)

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

- INPUT:
 - `l`: la lista dalla quale eliminare gli elementi
 - `e`: l'elemento da eliminare
- Comportamento: La funzione elimina tutte le istanze dell'elemento `e` dalla lista `l`. Se l'elemento non è presente, non fa nulla.