# Esame di Programmazione IP a.a. 2023/24 - Prova del 18 Giugno 2024

Prima di cominciare lo svolgimento, leggete attentamente tutto il testo. Questa prova è organizzata in due sezioni, ciascuna delle quali contiene diversi esercizi. Per ogni esercizio dovete progettare e implementare le funzioni richieste nei file specificati. Non è consentito modificare il nome delle funzioni, il tipo restituito, o i parametri. Potete realizzare altre funzioni ausiliarie se ritenuto necessario.

Potete utilizzare tutti gli #include che vi servono oltre a quelli già presenti. L'uso di funzioni di libreria standard per evitare di scrivere codice richiesto viene considerato un errore (esempio: l'uso di std::sort() se viene richiesto di implementare un algoritmo di ordinamento).

#### Sezione 1 - Array

### Sezione 1 - Array (Max 7.5 punti)

Per questa sezione lavorate nella cartella Sezione1. Per ogni esercizio, scrivete una funzione nel file specificato, completandolo secondo le indicazioni.

#### Materiale dato:

- array.h contenente le definizioni di tipo e le intestazioni delle funzioni (non modificare).
- mainTestArray.cpp contenente un main da usare per fare testing (non modificare).
- Es1-Funzione1.cpp, Es1-Funzione2.cpp, Es1-Funzione3.cpp da completare.

```
#### Es1-Funzione1 - (2.5 punti)
```

```cpp

bool arrayContainsArithmeticSeries(const int\* arr, unsigned int size, int difference);

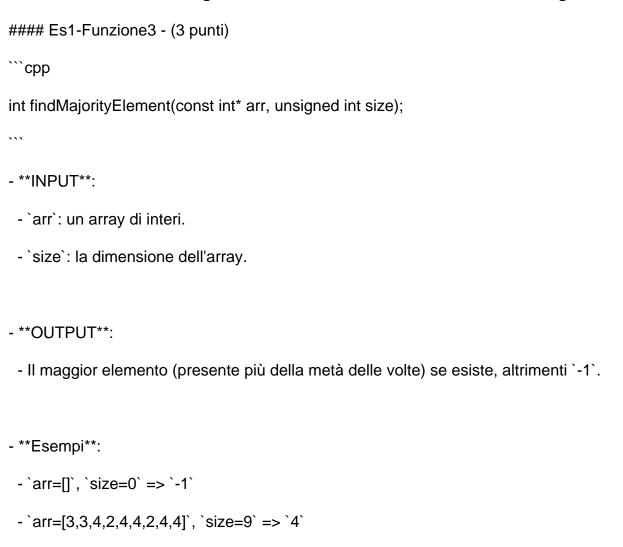
.....

- \*\*INPUT\*\*:
  - `arr`: un array di interi.
  - `size`: la dimensione dell'array.
  - `difference`: la differenza comune della serie aritmetica.

#### - \*\*OUTPUT\*\*:

- `TRUE` se l'array contiene una serie aritmetica con la differenza comune specificata, o se l'array è vuoto.
  - `FALSE` altrimenti.

```
- **Esempi**:
 - `arr=[]`, `size=0`, `difference=1` => `TRUE`
 - `arr=[2, 4, 6, 8]`, `size=4`, `difference=2` => `TRUE`
 - `arr=[1, 3, 7, 9]`, `size=4`, `difference=2` => `FALSE`
#### Es1-Funzione2 - (2 punti)
```cpp
int findSecondLargest(const int* arr, unsigned int size);
- **INPUT**:
 - `arr`: un array di interi.
 - `size`: la dimensione dell'array.
- **OUTPUT**:
 - Il secondo elemento più grande nell'array se esiste, altrimenti `-1`.
- **Comportamento**:
 - La funzione trova il secondo elemento più grande nell'array `arr`.
- **Esempi**:
 - `arr=[]`, `size=0` => `-1`
 - `arr=[3]`, `size=1` => `-1`
 - `arr=[3,1,4,2]`, `size=4` => `3`
 - `arr=[7,7,7,7]`, `size=4` => `-1` (non ci sono elementi distinti)
```



- `arr=[3,3,4,2,4,4,2,4]`, `size=8` => `-1`

#### Sezione 2 - Liste

```
### Sezione 2 - Liste (Max 8.5 punti)
```

Per questa sezione lavorate nella cartella Sezione2. Per ogni esercizio, scrivete una funzione nel file specificato, completandolo secondo le indicazioni.

```
Definizioni:

""cpp

typedef std::string Elem;

struct Cell {

Elem elem;

struct Cell* next;
};

typedef Cell* List;

""

Materiale dato:
```

- list.h contenente le definizioni di tipo e le intestazioni delle funzioni (non modificare).
- mainTestList.cpp contenente un main da usare per fare testing (non modificare).
- Es2-Funzione1.cpp, Es2-Funzione2.cpp, Es2-Funzione3.cpp da completare.

```
#### Es2-Funzione1 - (2 punti)
```cpp
```

unsigned int countOccurrences(const List &I, Elem s);
- **INPUT**:
- `l`: la lista nella quale contare le occorrenze.
- `s`: l'elemento di cui contare le occorrenze.
- **OUTPUT**:
- Il numero di occorrenze di `s` nella lista `l`.
#### Es2-Funzione2 - (3 punti)
```cpp
bool insertElemInSortedOrder(List &I, Elem s);
- **INPUT**:
- `l`: la lista nella quale inserire l'elemento.
- `s`: l'elemento da inserire nella lista.
- **OUTPUT**:
- `TRUE` se l'inserimento va a buon fine.
- `FALSE` altrimenti (in caso di errore).
- **Comportamento**:
- La funzione inserisce l'elemento `s` nella lista `l` mantenendo l'ordine crescente.
#### Es2-Funzione3 - (3.5 punti)

```cpp
void removeDuplicates(List &I);
- **INPUT**:
- `l`: la lista dalla quale rimuovere i duplicati.
- **Comportamento**:
- La funzione rimuove tutti i duplicati dalla lista `l`, mantenendo solo la prima occorrenza di ciascun

elemento.