

# Esame di Programmazione (mod A) - CdL AIDA

V Appello Settembre 2021

Giulio Caravagna ([gcaravagna@units.it](mailto:gcaravagna@units.it))

## 1 ISTRUZIONI

L'appello contiene **6** esercizi (A1, A2, A3, B1, B2, B3) da risolvere in 3 ore. Il template si trova su Moodle in formato ZIP, su Moodle dovete carica la vostra soluzione.

**Importante.** A1, A2 e A3 sono di *sbarramento* e permettono di raggiungere 18/30. B1, B2 e B3 valgono fino al raggiungimento del voto massimo di 30/30.

**Risoluzione degli esercizi di sbarramento.** Usando `repl.it`, risolvete l'esercizio partendo dal file `main.c` e testate il codice con i comandi `make test1`, `make test2` e `make test3`. Prima di ogni test ricordatevi di digitare `make clean`.

## 2 ESERCIZI DI SBARRAMENTO (18 PUNTI)

**A1.** Si scriva una funzione *iterativa* `overlap_size` che prenda in input 4 interi positivi  $i, j, t, u$  tali per cui  $i < j$ ,  $t < u$  e  $j \geq t$ . I 4 interi definiscono due intervalli di numeri naturali

$$[i, j] \quad e \quad [t, u]$$

per cui si vuole calcolare la dimensione dell'intersezione  $I = |[i, j] \cap [t, u]|$ . Per esempio, gli intervalli  $[1, 5]$  e  $[3, 12]$  hanno sovrapposizione  $[3, 5] = \{3, 4, 5\}$  e quindi dimensione 3.

Il calcolo della dimensione *deve essere fatto iterativamente* utilizzando una funzione `is_inside(x, y, z)` che restituisce 0 se  $x \in [y, z]$ , e -1 altrimenti.

**A2.** Si scriva un programma C *iterativo* che calcoli, per un dato  $n \geq 1$  in input, la successione di interi

$$\begin{cases} a_1 = 0 \\ a_2 = 0 \\ a_n = (a_{n-2} + a_{n-1} - 1)(n + 1) & \text{con } n \geq 3 \text{ se } a_{n-1} > 2a_{n-2} \\ a_n = (a_{n-2} - a_{n-1} + 1)(a_{n-2} + 1)^2 & \text{con } n \geq 3 \text{ altrimenti.} \end{cases}$$

**A3.** Si consideri la successione

$$F_0 = 1 \quad F_1 = 100 \quad F_n = nF_{n-1} - \frac{nF_{n-2}}{2} \quad n \geq 2$$

e la quantità  $\mathbf{F} = \sum_{i=1}^N F_{x_i}$  calcolata a partire da un *array*  $\mathbf{x}$  di  $N$  valori non negativi  $x_1, x_2, \dots, x_N$ .

Si scriva un programma C che dato  $\mathbf{x}$  calcoli  $\mathbf{F}$  *iterativamente* e ogni  $F_i$  *ricorsivamente*. Per esempio, se  $\mathbf{x} = [1, 2, 0]$  allora  $\mathbf{F} = F_1 + F_2 + F_1$ ;  $F_1$ ,  $F_2$  ed  $F_1$  sono ricorsive, il prodotto iterativo.

*Suggerimento:* Si consideri che  $F_n$  sono numeri con la virgola (**double**).

### 3 ESERCIZI OPZIONALI

#### 3.1 Es. B1 (6 punti)

In C, si vogliono definire *liste linkate* che possono memorizza un *array di interi* in ciascun loro elemento; si desidera inoltre permettere agli array di avere dimensione variabile, e.g., una lista potrebbe essere

```
[{1,2,3}] --> [{9}] --> [{43, 5}] // array di 3, 1 e 2 elementi
```

Si usi il seguente template per definire la **struct** necessaria ad implementare la lista.

```
// struttura
struct elemento{

    // dato memorizzato
    ...

    // puntatore
    struct elemento * next;
};

// tipi
typedef struct elemento ElementoDiLista;
typedef ElementoDiLista * ListaDiElementi;
```

Si definiscano, secondo la **struct** sopra definita, le funzioni

```
int ntot(ListaDiElementi lista)
int largest(ListaDiElementi lista)
```

dove *i*) **ntot** restituisce il numero totale degli elementi inseriti nella lista (somma del numero di elementi in ciascun array contenuto), e *ii*) **largest** che restituisce il numero massimo di elementi contenuti in un elemento delle lista. Ad esempio (dopo la **init**)

```
// supponendo si crei una lista di un singolo array con 4 elementi, usando
// un metodo "init"
ListaDiElementi list = init(4);

// si aggiunge un secondo elemento, stavolta con un array di 12 elementi
list->next = init(12);

// a questo punto avremmo
// - ntot(list) che restituisce 12 + 4 = 16
// - largest(list) che restituisce 12
```

## 3.2 Es. B2 (3 punti)

Si consideri questo programma C

```
// funzione ausiliaria
int f(int x){
    return(x+6); // A
}

int x = 6;
int y = x + 1; // B

y = f(y);
int * z = y;

z = f(*z - y) // C
```

Si rappresenti la memoria del programma ai punto A, B e C. Si noti che A viene eseguito 2 volte.

## 3.3 Es. B3 (3 punti)

- Si scriva una classe **Rettangolo** per costruire un rettangolo con lunghezza ed altezza.
- Si crei un metodo **Perimetro()/Area()** per calcolare il perimetro/area del rettangolo.
- Si crei un metodo **display()** che mostri lunghezza, altezza, perimetro ed area per un oggetto di classe **Rettangolo**.
- Si crei una sottoclasse **Parallelepipedo** che erediti da **Rettangolo** e contenga un attributo altezza ed un metodo **Volume()** per il calcolo del volume del parallelepipedo.