Esame di Programmazione (mod A) - CdL AIDA

III Appello Giugno 2021

Giulio Caravagna (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 (`Appello.zip`), dove dovete carica la vostra soluzione. In caso di problemi con Moodle, inviate le soluzioni via mail.

Importante. `A1`, `A2` e `A3` sono di *sbarramento* e permettono di raggiungere 18/30, il minimo per superare l'esame. `B1`, `B2` e `B3` valgono fino al raggiungimento del voto massimo di 30/30.

Risoluzione degli esercizi di sbarramento. Risolvete l'esercizio partendo dal file `main.c`. Potete usare `repl.it` trascinando la cartella template, ed esporatando la soluzione finale, testate con i comandi `make test1`, `make test2` e `make test3`. Le cartelle contengono inputs (`input`), i risultati attesi (`result`) ed i vostri risultati (`output`). Una soluzione senza output non viene considerata valida.

2 Esercizi di sbarramento (18 punti)

A1. Si scriva una funzione che prenda in input un numero $c_0 c_1 \dots c_k$ e restitutisca il numero $c_k c_{k-1} \dots c_0$ con le cifre c_i in ordine inverso (per esempio dato `2134` restituisce `4312`).

Suggerimento: Si consideri che $2134 = 2 \times 1000 + 1 \times 100 + 3 \times 10 + 4 \times 1$ ed un calcolo analogo vale per il suo inverso.

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

$$\begin{cases} a_1 = 2 \\ a_2 = 1 \\ a_n = (n - a_{n-1})a_{n-2} \quad \text{con } n \geq 3 \quad \text{se } a_{n-1} \text{ e' pari} \\ a_n = (n + a_{n-2})a_{n-1} \quad \text{con } n \geq 3 \quad \text{se } a_{n-1} \text{ e' dispari.} \end{cases}$$

A3. Si consideri la successione di Fibonacci $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \dots$ definita per ricorsione

$$F_n = F_{n-1} + F_{n-2}, \quad F_0 = 0, \quad F_1 = 1.$$

Si scriva un programma C che, dato un *array di valori non negativi* x_1, ..., x_N , calcoli *ricorsivamente*

$$\boldsymbol{F} = \sum_{i=1}^{N} F_{x_i} \,,$$

dove il termine per l'(x_i)-esimo numero di Fibonacci (F_{x_i}) viene calcolato in modo *iterativo*. Per esempio, se $\boldsymbol{x} = [1, 2, 0]$ allora $\boldsymbol{F} = F_1 + F_2 + F_0 = 1 + 1 + 0 = 2$; F_0, F_1 ed F_2 sono iterative, la somma ricorsiva.

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 sequente template per definire la `struct` necessaria ad implementare la lista.

```c
// 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

```c
ListaDiElementi init(int n)
void print(ListaDiElementi lista)
```

dove *i*) `init` costruisce una lista di un singolo elemento, il quale contiene un array di $n > 0$ elementi , e *ii*) `print` stampa *ricorsivamente* la lista, mostrando ad esempio (dopo la `init`)

```c
ListaDiElementi list = init(4);

print_list(list);
// -> n = 4 | 0, 0, 0, 0,

list->next = init(12);

print_list(list);
// -> n = 4 | 0, 0, 0, 0,
// -> n = 12 | 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, %
```

Suggerimento. Se gli array hanno dimensione variabile, la gestione della memoria deve essere esplicita.

3.2 Es. B2 (3 punti)

Si consideri questo programma C

```
int x = 6;
int *y = &x;

for(int i = 2; i < *y; i++)
{
  # A
  if(i %2 != 0)
  {
    int x = -1;
    *y = x;
  }
  else
  {
    x = x + *y;
  }
}
```

Si rappresenti la memoria del programma al punto `A` per ciascun ciclo di esecuzione del `for`. La memoria finale di questo programma sarebbe equivalente inizializzando `i=3` o `i=4`?

3.3 Es. B3 (3 punti)

Si consideri la classe Python che descrive un veicolo con velocita massima `v`, kilometraggio `km` e posti `p`.

```
class Veicolo:
    def __init__(self, p, v, km):
        self.posti = p
        self.velocita = self.v
        self.km = km
```

Specificare 4 sottoclassi che ereditano da `Veicolo`:

- `Auto4` e `Auto5`, per le auto a `p=4` e a `p=5` posti;
- `Bus_90` e `Bus_130`: per autobus la cui velocita massima e' `v=90` e `v=130`.