Esame di Programmazione (mod A) - CdL AIDA

VI Appello Settembe 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 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

$$I = [i, j] \quad e \quad J = [t, u]$$

per cui si vuole calcolare la dimensione dell'insieme $\bar{I} = |\{x \in I, x \notin J\} \cup \{x \notin I, x \in J\}|$. Per esempio, gli intervalli $[1, 5]$ e $[3, 12]$ hanno sovrapposizione $\bar{I} = \{1, 2\} \cup \{6, 7, 8, 9, 10, 11, 12\}$ e quindi dimensione 9.

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 che, per un dato numero $n = c_0 c_1 \dots c_k$ composto da k cifre $c_i \in 0, \dots, 9$, calcoli il prodotto delle cifre

$$s_n = \prod_{i=0}^{k} c_i$$

in modo *ricorsivo*.

Per esempio:

- se $n = 1234$ allora $s_n = 24$;
- se $n = 2435$ allora $s_n = 120$;

A3. Si consideri la successione

$$F_0 = 1 \qquad F_1 = 12 \qquad F_n = 3(F_{n-1} - F_{n-2}) + (F_{n-2} - F_{n-1})(n - F_1) \qquad n \geq 2$$

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

Si scriva un programma C che dato $\boldsymbol{x}$ calcoli $\boldsymbol{F}$ *iterativamente* e ogni F_i *ricorsivamente*. Per esempio, se $\boldsymbol{x} = [1, 2, 0]$ allora $\boldsymbol{F} = F_1 + F_2 + F_1$; F_1, F_2 ed F_3 sono ricorsive, il prodotto iterativo.

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

3 Esercizi opzionali

3.1 Es. B1 (5 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
int ntot(ListaDiElementi lista)
int largest(ListaDiElementi lista)
```

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

```c
// 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 (4 punti)

Si consideri questo frammento di codice C

```c
int main(void) {
  int i = 1;
  int a[3];
  int j = i + 8; // P1

  for(int i = 0; i < 3; i++)
  {
    a[i] = i*i;
    j = i + 1; //P2 (terza iterazione)
  }

  j = i - j + 11; // P3
}
```

Si disegni la memoria del programma al punto P1, P2 (specificatamente alla terza iterazione del ciclo), ed al punto P3 (fine del `main`).

3.3 Es. B3 (3 punti)

Si considerino le seguenti classi

```python
class C1:
  def __init__(self):
    self.i = 1

class C2(C1):
  def __init__(self):
    super().__init__()
    self.i = 1
```

- Si riscriva la gerarchia di classi in modo che il costruttore di `C1` prenda un parametro generico (generalizzando l'assegnamneto `i=1`), e si cambi di conseguenza `C2`.
- Si scriva un metodo `quadrato` che restituisca il valore del quadrato del numero memorizzato; si decida e motivi in quale classe tale metodo debba essere realizzato.