# Esame Completo di Programmazione I - BioInformatica

1 febbraio 2019 (tempo disponibile: 2 ore)

## Esercizio 1 (6 punti)

Si scriva un programma `sum.c` che implementa una funzione `int sum(int arr[], int length)`. Tale funzione deve ricevere un array `arr` di interi, lungo `length`, e deve restituire la somma degli elementi di `arr` che abbiano l'elemento precedente dispari. Per esempio, se `arr` fosse $\{2, 3, 4, 1, 5\}$, la funzione dovrebbe restituire 9 (la somma di 4 e 5). Si scriva il file di header `sum.h` in cui si dichiara tale funzione.

## Esercizio 2 (8 punti)

Si scriva un programma `main_sum.c` che include la funzione dell'Esercizio 1 tramite il file di header `sum.h`. Il programma `main_sum.c` deve contenere una funzione iniziale `main` che esegue le seguenti operazioni:

1. legge da tastiera la lunghezza `length` di un array, richiedendola ad oltranza se fosse inserita negativa;
2. crea un array `elements` di `length` interi;
3. legge da tastiera gli elementi di tale array, uno alla volta;
4. chiama la funzione `sum` dell'Esercizio 1, passando `elements` e `length`;
5. stampa sul video il risultato di tale chiamata.

## Esercizio 3 (10 punti)

Si definisca una struttura `trainingRecorder` che implementa un registro per memorizzare risultati di allenamenti podistici. Si scrivano i file `trainingRecorder.h` e `trainingRecorder.c` che implementano le funzioni:

- `struct trainingRecorder *constructTrainingRecorder(char *trainingName)`, che restituisce un nuovo registro vuoto con il nome indicato;
- `void destructTrainingRecorder(struct trainingRecorder *this)`, che dealloca `this`;
- `void addRun(struct trainingRecorder *this, float length, float time)`, che aggiunge un nuovo allenamento (*run*) al registro memorizzandone la lunghezza del percorso (in chilometri) e il tempo impiegato (in minuti). Un numero illimitato di allenamenti possono essere aggiunti ad un registro;
- `float averageSpeed(struct trainingRecorder *this)`, che restituisce la velocità media degli allenamenti inseriti in `this`; se `this` non contenesse nessun allenamento, questa funzione deve restituire 0.0;
- `char *toString(struct trainingRecorder *this)`, che restituisce una nuova stringa ottenuta dal nome del registro `this` seguito dalla velocità media e dal numero degli allenamenti in esso contenuti;

- `int compareAverageSpeeds(struct trainingRecorder *tr1, struct trainingRecorder *tr2)`, che confronta `tr1` e `tr2` e restituisce -1 se la velocità media del primo è minore di quella del secondo, 1 se la velocità media del primo è maggiore di quella del secondo, 0 altrimenti.

Se tutto è corretto, l'esecuzione del seguente programma `main_trainingRecorder.c`:

```
#include <stdlib.h>
#include <stdio.h>
#include "trainingRecorder.h"

int main(void) {
  struct trainingRecorder *tr1 = constructTrainingRecorder("New York Marathon");
  struct trainingRecorder *tr2 = constructTrainingRecorder("Rome Marathon");
  char *s;

  addRun(tr1, 10, 51);
  addRun(tr1, 15, 79);
  addRun(tr1, 20, 120);

  addRun(tr2, 8, 40);
  addRun(tr2, 35, 180);
  addRun(tr2, 20, 115);
  addRun(tr2, 40, 195);

  printf("%s\n", s = toString(tr1));
  free(s);
  printf("%s\n", s = toString(tr2));
  free(s);
  printf("Comparison: %d\n", compareAverageSpeeds(tr1, tr2));

  destructTrainingRecorder(tr1); destructTrainingRecorder(tr2);
  return 0;
}
```

deve stampare:

```
Training name: New York Marathon; average speed:  10.80 Km/h; number of runs: 3
Training name: Rome Marathon; average speed:  11.66 Km/h; number of runs: 4
Comparison:  -1
```

## Esercizio 4 (8 punti)

Si modifichi il programma `coppia.c` definendo le funzioni `init`, `f`, `g` ed `h`, in modo che alla fine compili senza errori e la sua esecuzione stampi:

```
Il valore della coppia e' uguale a (5,10)
Il valore della coppia e' uguale a (45,10)
Il valore della coppia e' uguale a (45,10)
Il valore della coppia e' uguale a (45,12)
```

Non potete modificare altro in `coppia.c`; quindi, per esempio, non potete modificare la funzione `main` o la `struct coppia`.