NOTE SULLO SVOLGIMENTO DELLA PROVA:

- PER SVOLGERE L'ELABORATO, APRIRE il Dev-C++ (dal Menù **`Avvio`** (o **`Start`**) nella barra degli strumenti in fondo allo schermo, selezionare Programmi e quindi Dev-C++);
- PRIMA DI INIZIARE LO SVOLGIMENTO DELL'ELABORATO, selezionare la voce **`Identifica studente`** nel menù **`Strumenti`** all'interno dell'ambiente Dev-C++ e inserire i dati richiesti;
- per svolgere l'elaborato, aprire il progetto *esaInf.dev* presente nel directory *c:\esame\esaInf* e scrivere le funzioni richieste nel file *compito.cpp*, già presente nel progetto;
- per una corretta stampa dell'elaborato bisogna mantenere il codice entro i margini imposti dall'ambiente Dev-C++ (linea verticale presente alla destra della pagina).

Palestra

Le attività e gli iscritti di una palestra possono essere rappresentati attraverso le seguenti costanti e strutture:

```
const int MAXN = 100;
const int MAXA = 5;
const int MAXI = 200;

enum Attivita {PESI, CORPO_LIBERO, PILATES, KARATE, SPINNING};

struct Iscritto {
  char nome[MAXN];
  char cognome[MAXN];
  float credito;
  Attivita att[MAXA];
  int numAtt;
};

struct Palestra {
  Iscritto iscr[MAXI];
  int numIscr;
};
```

La struttura `Iscritto` contiene i dati di un iscritto alla palestra: il significato dei campi `nome` e `cognome` è palese; il campo `credito` contiene il credito corrente dell'iscritto; il campo `att` specifica le attività che tale iscritto pratica; il campo `numAtt` indica il numero di attività che tale iscritto pratica. La struttura `Palestra` è composta da due campi: un array `iscr` che contiene i dati degli iscritti, e il campo `numIscr` che indica il numero di iscritti effettivamente presenti.

Scrivere il corpo delle seguenti funzioni C++.

1. **`bool nuovoIscritto(Palestra* pp, const char n[], const char c[], float s)`** che inserisce nella palestra un nuovo iscritto avente nome `n` e cognome `c`. Il parametro `s` specifica il valore del credito iniziale del nuovo iscritto. Il nuovo iscritto non pratica nessuna attività. La funzione restituisce *true* se l'inserimento va a buon fine, *false* altrimenti.

2. **`bool aggiungiAttivita(Palestra* pp, const char n[], const char c[], Attivita a)`** che aggiunge l'attività `a` a quelle che possono essere praticate dall'iscritto avente nome `n` e cognome `c` (che deve essere già presente nella palestra). Si supponga che non ci siano casi di omonimia. La funzione restituisce *true* se l'attività `a` viene aggiunta, *false* altrimenti.

3. **`bool salvaOrd(const Palestra* pp, const char filen[])`** che salva nel file avente nome `filen` la lista degli iscritti secondo il seguente formato (nel file vengono riportati solo i campi credito, nome e cognome):
   45.4 Mario Rossi
   67.0 Giuseppe Verdi
   115.5 Sara Gialli
   ...
   La lista degli iscritti deve essere ordinata per valori crescenti del campo credito. La funzione restituisce *true* se il salvataggio va a buon fine, *false* altrimenti.

4. **`void prelevaRata(Palestra* pp, Attivita a, float r)`** che sottrae la quantità `r` al credito di tutti gli iscritti della palestra che praticano l'attività `a`. Il credito di un iscritto può diventare negativo.

5. **`bool elimina(Palestra* pp, const char n[], const char c[])`** che elimina dalla palestra l'iscritto avente nome `n` e cognome `c`. Si supponga che non ci siano casi di omonimia. La funzione restituisce *true* se l'eliminazione va a buon fine, *false* altrimenti.

6. **`int quantiRosso(const Palestra* pp)`** che restituisce il numero di iscritti aventi un credito negativo.