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; se tale voce non è presente, proseguire con il punto successivo (l'identificazione verrà eseguita manualmente dal docente).
- 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).

Richieste alloggio

Le richieste per alloggi studenti possono essere gestite attraverso le seguenti costanti e strutture:

```
const int MAXRIC = 100;
const int MAXS = 100;

enum Tipo {Singola, Doppia};

struct Richiesta {
  char nome[MAXS];
  char cognome[MAXS];
  Tipo tipo;
  int punti;
};

struct InsiemeRichieste {
  Richiesta ric[MAXRIC];
  int numRich;
};
```

La struttura `Richiesta` si compone di: un campo `nome` che contiene il nome dello studente che chiede un alloggio, il campo `cognome` che contiene il cognome dello studente stesso, un campo `tipo` che indica il tipo di alloggio richiesto (`Singola` o `Doppia`) e infine un campo `punti` che determinerà la sua posizione in graduatoria. La struttura `InsiemeRichieste` è composta da due campi: un array `ric` che contiene i dati delle richieste, e il campo `numRich` che indica il numero di richieste effettivamente presenti.

Scrivere il corpo delle seguenti funzioni C++.

1. **`bool inserisci(InsiemeRichieste* pi, const char no[], const char co[], Tipo ti, bool fu, float re, float me)`** che inserisce nell'insieme una nuova richiesta di alloggio. La nuova richiesta può essere inserita solo se c'è spazio disponibile nell'insieme. I parametri `no` e `co` indicano rispettivamente il nome e il cognome dello studente che fa la richiesta. Il parametro `ti` indica se lo studente vuole una camera singola o una doppia. Il parametro `fu` indica se lo studente è fuorisede (*true*) oppure no (*false*). Il parametro `re` indica il reddito della sua famiglia. Il parametro `me`

indica la media dei voti dello studente. I parametri `fu`, `re` e `me` vengono usati per calcolare i punti da associare alla richiesta. In particolare:

- Se lo studente è fuorisede vengono attribuiti 5 punti, 0 se non lo è.
- Se il reddito della famiglia è minore di 10000 vengono attribuiti 5 punti, se il reddito della famiglia è maggiore o uguale a 10000 ma minore di 20000 vengono attribuiti 2 punti, se il reddito è maggiore di 20000 non vengono attribuiti punti.
- Per quanto riguarda la media, vengono attribuiti punti in quantità pari alla differenza tra la media arrotondata all'intero più vicino e 18 (per esempio, se lo studente ha come media 23.3 gli verranno attribuiti 5 punti).

Quando una nuova richiesta viene inserita nell'array ric, questa deve essere inserita in modo tale da lasciare tale array ordinato per valori decrescenti del campo punti. La funzione restituisce *true* se l'inserimento va a buon fine, *false* altrimenti. Suggerimento: utilizzare una funzione separata per il calcolo dei punti di una richiesta.

2. **`bool alloggioDisponibile(InsiemeRichieste* pi, Tipo ti)`** che elimina dall'insieme delle richieste quella che ha il maggior numero di punti e che corrisponde al tipo indicato da `ti`. La rimozione di una richiesta implica un ricompattamento del contenuto dell'array `ric`, avendo cura di non perturbare l'ordinamento delle richieste. La funzione restituisce `true` se una richiesta viene rimossa, `false` altrimenti.

3. **`void quantePerTipo(const InsiemeRichieste* pi, int* psing, int* pdop)`** che conta il numero di richieste relative a singole e a doppie presenti nell'insieme. Il numero di richieste relative a singole viene restituito al chiamante attraverso l'oggetto puntato da `psing`, il numero di richieste relative a doppie viene restituito al chiamante attraverso l'oggetto puntato da `pdop`.

4. **`bool salvaOrdNome(const InsiemeRichieste* pi, const char nf[])`** che salva nel file di nome `nf` i dati delle richieste presenti nell'insieme. Le richieste devono essere ordinate per valori crescenti del campo cognome. Il contenuto del file deve essere analogo al seguente esempio:

```
bianchi alessio 5    Singola
rossi mario     22   Doppia
verdi carlo     5    Singola
```

5. **`bool omonimia(const InsiemeRichieste* pi)`** che restituisce *true* se nell'insieme sono presenti due richieste aventi lo stesso nome e lo stesso cognome. In caso contrario restituisce *false*.