Un sistema per la gestione di dati relativi a calciatori può essere realizzato attraverso le seguenti costanti e strutture:

```
const int MAXNUM = 100;
const int MAXL = 50;

enum Ruolo {PORTIERE, DIFENSORE, CENTROCAMPISTA, ATTACCANTE};

struct Data {
  int giorno;
  int mese;
  int anno;
};

struct Giocatore {
  char nome[MAXL];
  char cognome[MAXL];
  Ruolo ru;
  Data sca;
  char squadra[MAXL];
  double quot;
};

struct ArchivioGio {
  Giocatore gio[MAXNUM];
  int numGio;
};
```

con **MAXNUM** il numero massimo di giocatori che possono essere gestiti e **numGio** il numero di giocatori effettivamente presenti. Ogni giocatore è caratterizzato da: due stringhe, **nome** e **cognome**, di al più **MAXL** caratteri, un campo **ru** (di tipo **Ruolo**) che rappresenta il tipo di ruolo ricoperto, un campo **sca** (di tipo **Data**) che indica la data di scadenza del contratto, una stringa (di lunghezza massima **MAXL**) **squadra** che indica la squadra di appartenenza, e un campo **quot** di tipo reale che indica la quotazione di mercato del giocatore (in milioni di euro).

Scrivere il corpo delle seguenti funzioni C++.

1. **double quoMedia(const ArchivioGio* pa, Ruolo r)** che restituisce il valore medio della quotazione di mercato dei giocatori presenti in archivio aventi il ruolo indicato da **r**.
2. **int inScadenza(const ArchivioGio* pa, Data d, Giocatore vet[])** che copia in **vet** i dati dei giocatori presenti in archivio che hanno un contratto che scade prima della data specificata da **d**; la funzione restituisce il numero di oggetti **Giocatore** inseriti in **vet**.
3. **void quanti(const ArchivioGio* pa, int q[4])** che conta il numero di giocatori presenti a seconda dei vari ruoli e restituisce il risultato attraverso l'array **q** (nel primo elemento di **q**

viene riportato il numero di giocatori di tipo **PORTIERE**, nel secondo elemento il numero di giocatori di tipo **DIFENSORE**, e così via).

4. **`void salvaOrd(const ArchivioGio* pa, const char nf[])`** che salva nel file il cui nome è specificato da **`nf`**, i campi **`nome`**, **`cognome`** e **`quot`** dei giocatori presenti in archivio ordinandoli per quotazione di mercato decrescente.
5. **`int elimina(ArchivioGio* pa, const char s[])`** che elimina dall'archivio tutti i giocatori che appartengono alla squadra **`s`**; la funzione restituisce il numero di giocatori eliminati dall'archivio.

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);
- SE L'ELABORATO È STATO COMPILATO SENZA ERRORI, PRIMA DELLA CONSEGNA, selezionare la voce **`Consegna Compito`** nel menù **`Strumenti`** all'interno dell'ambiente Dev-C++ e premere il tasto INVIO fino a quando non sparisce la finestra che è stata attivata.