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).

Supponiamo che un ospedale sia composto da una accettazione e da un insieme di reparti, tutti in grado di contenere pazienti. Supponiamo inoltre che l'accettazione, e solo questa, operi come una coda. Quando un paziente viene "accettato", viene inserito in fondo alla coda. Il paziente che si trova in testa alla coda può essere spostato nel suo reparto di destinazione, a patto che il suo reparto non sia ancora "da assegnare". In quest'ultima circostanza, il paziente viene rimesso nuovamente in fondo alla coda accettazione. Un paziente può inoltre essere trasferito da un reparto all'altro.

Il sistema di gestione può essere realizzato attraverso le seguenti costanti e strutture:

```
const int MAXL = 50;
const int DIMCOD = 20;
const int DIMREP = 10;

enum TipoRep {PEDIATRIA, ORTOPEDIA, MED_INTERNA, RADIOLOGIA, DA_ASSEGNARE};

struct Paziente {
  char nome[MAXL];
  char cognome[MAXL];
  TipoRep destinaz;
};

struct CodaPaz {
  Paziente paz[DIMCOD];
  int front, back;
};

struct Reparto {
  Paziente paz[DIMREP];
  int numPaz;
};

struct Ospedale {
  CodaPaz accettazione;
  Reparto rep[RADIOLOGIA+1];
};
```

La struttura `Paziente` contiene il nome e il cognome di un paziente (stringhe di lunghezza minore di `MAXL` caratteri) e il tipo del reparto di destinazione. Un reparto è rappresentato da una istanza della struttura `Reparto`. In particolare, il campo `paz` viene utilizzato per contenere i dati dei pazienti presenti in tale reparto e il campo `numPaz` contiene il numero di pazienti effettivamente presenti (in un reparto possono essere conentuti al più `DIMREP` pazienti). La struttura `CodaPaz` rappresenta una coda di pazienti, memorizzati all'interno del suo campo `paz` (un array di dimensione `DIMCOD`). Gli indici `front` e `back` vengono utilizzati per indicare rispettivamente le posizioni di estrazione e inserimento. Un ospedale è rappresentato da una istanza della struttura `Ospedale`. Il campo `accettazione` della struttura è una coda di pazienti, mentre il campo `rep` rappresenta l'insieme dei reparti dell'ospedale. All'interno dell'array `rep`, il tipo del reparto che occupa la posizione i-esima è pari al valore dell'i-esimo enumeratore di `TipoRep` (l'elemento di indice *0* di `rep` rappresenta il reparto pediatria, quello di indice *1* il reparto ortopedia, e così via).
Inizialmente l'ospedale è vuoto: non ci sono pazienti in accettazione e nei reparti.

Scrivere il corpo delle seguenti funzioni C++.

1. **`bool accetta(Ospedale* po, const char n[], const char c[], TipoRep r)`** che inserisce nella coda `accettazione` dell'ospedale puntato da `po` un nuovo paziente, avente nome `n`, cognome `c`, e destinazione pari a `r`. La funzione restituisce *true* se l'inseriemento va a buon fine, *false* altrimenti (la coda è piena).
2. **`bool sposta(Ospedale* po)`** che, se possibile, sposta il paziente che si trova in testa alla coda `accettazione` dell'ospedale puntato da `po` nel suo reparto di destinazione. Affinché un paziente possa essere spostato nel suo reparto devono sussistere le due seguenti condizioni: *i)* c'è spazio nel reparto di destinazione, *ii)* al paziente è già stato assegnato un reparto. Se lo spostamento ha luogo viene restituito il valore *true* al chiamante. Nel caso in cui invece lo spostamento non possa vere luogo, il paziente viene rimesso in fondo alla coda e viene restituito il valore *false* al chiamante.
3. **`bool trasferisci(Ospedale* po, const char n[], const char c[], TipoRep da, TipoRep a)`** che trasferisce un paziente dal reparto `da` al reparto `a`. Il paziente da trasferire è quello con nome `n` e cognome `c` (si supponga che non vi siano casi di omonimia). La funzione restituisce il valore *true* se il trasferimento ha luogo, *false* altrimenti (il trasferimento non può aver luogo se si verifica almeno una delle seguenti condizioni: *i)* non c'è un paziente con nome e cognome specificati, *ii)* non c'è spazio nel reparto di destinazione, *iii)* almeno uno degli argomenti `da` e `a` non identifica un reparto valido).
4. **`int quantiAffollato(const Ospedale* po, TipoRep* pq)`** che restituisce il numero di pazienti presenti nel reparto più affollato. Oltre a restituire tale numero, la funzione riempie l'oggetto puntato da `pq` con l'indicazione del tipo di reparto in questione.
5. **`bool salva(const Ospedale* po, const char nf[])`** che salva nel file di nome `nf` l'elenco di tutti i pazienti presenti nell'ospedale (accettazione più reparti). L'elenco è ordinato alfabeticamente per cognome. La funzione restituisce il valore *true* se l'operazione va a buon fine, *false* altrimenti.