ESERCIZIO 1

Scrivete un programma per la gestione delle informazioni relative ai voli giornalieri in partenza da un certo aeroporto. Il programma deve leggere da file le informazioni relative ai voli, memorizzarle in opportune strutture di dati e consentire l'aggiornamento del numero di posti liberi di un determinato volo e la cancellazione di un volo una volta partito.
Di ciascun volo il file memorizza il numero, la destinazione, l'orario di partenza ed il numero dei posti liberi. Considerate, ad esempio, il seguente file `voli.dat`:

```
AZ111 milano       12:34  23
AZ222 buenos-aires 12:55 245
BA333 toronto       4:15  20
KM444 madrid       10:12 100
```

a) Progettate le strutture di dati adatte a mantenere le informazioni lette da file. In particolare utilizzate una lista realizzata tramite strutture e puntatori. Ogni struttura deve contenere le seguenti informazioni:
   - numero del volo (stringa di 5 caratteri senza spazi bianchi);
   - destinazione del volo (stringa di al più 20 caratteri senza spazi bianchi);
   - orario di partenza, espresso in ore e minuti (struttura a due campi interi);
   - numero di posti liberi (intero).

b) Scrivete una funzione che crei una lista di voli leggendo le informazioni da file. La funzione deve ricevere come parametro il nome di un file contenente le informazioni sui voli in partenza; leggere il contenuto del file e creare una lista di voli; restituire alla funzione chiamante il puntatore iniziale della lista creata.

c) Scrivete una funzione che, prendendo come parametro il puntatore iniziale della lista dei voli in partenza, stampi su standard output il numero, la destinazione, l'orario di partenza e il numero di posti liberi di ciascun volo.

d) Scrivete la funzione `main()` che attiva in modo opportuno le due funzioni precedenti.

e) Scrivete una funzione per l'aggiornamento del numero dei posti liberi di un determinato volo. La funzione deve ricevere come parametri il puntatore iniziale della lista dei voli, il numero del volo ed il nuovo valore da assegnare al numero dei posti liberi, ed aggiornare la lista in modo opportuno.

f) Scrivete una funzione che elimini un volo dalla lista dei voli in partenza. La funzione deve ricevere come parametri il puntatore iniziale della lista ed il numero del volo da eliminare, ed aggiornare la lista eliminando il volo.

```
#include <iostream>
#include <fstream>
#include <string.h>
#include <stdlib.h>

using namespace std;
struct tipo_orario {
     int hh;
     int mm;
};
struct volo {
     char numero[5+1];
     char dest[20+1];
      tipo_orario orario;
     int postiLiberi;
};

struct nodo {
     volo info;
     nodo* next;
};

nodo* creaNodo(volo t);
nodo* inserisciInTesta(nodo* L, nodo* tmp);

nodo* creaListaDaFile(char nomeFile[]);      // Soluzione domanda (b)

void stampaLista(nodo* L);                   // Soluzione domanda (c)

void aggiornaPostiLiberiVolo(nodo* L, char numVolo[], int posti);
                                             // Soluzione domanda (e)

lista eliminaVolo(lista L, char numVolo[]);  // Soluzione domanda (f)

int main( ) {
     char nomeFile[] = "voli.dat";
     nodo* listaVoli = NULL;

     listaVoli = creaListaDaFile(nomeFile);
     stampaLista(listaVoli);

     cout << "\n\n ++++ Aggiorno Volo ++++\n\n";
     aggiornaPostiLiberiVolo(listaVoli, "AZ222", 247);
     stampaLista(listaVoli);

     cout << "\n\n ++++ Elimino Volo ++++ \n\n";
     listaVoli = eliminaVolo(listaVoli, "AZ3111");
     stampaLista(listaVoli);

     return 0;
}
```

```
nodo* creaNodo(volo t){
      nodo* elemento = new nodo;
      elemento->info = t;
      elemento->next = NULL;
      return elemento;
}

nodo* inserisciInTesta(nodo* L, nodo* tmp){
     if ( L == NULL || tmp == NULL )
       return L;
     tmp->next = L;
      return tmp;
}

void stampaLista(nodo* L){
      if ( L != NULL ) {
            volo tmp = L->info;
            cout << " Numero -> " << tmp.numero;
            cout << " \n Destinazione -> " << tmp.dest;
            cout << " \n Orario -> " << tmp.hh << ":" << tmp.mm;
            cout << " \n Posti liberi -> " << tmp.postiLiberi << "\n\n";
            stampaLista(L->next);
      }
}

lista creaListaDaFile(char nomeFile[]) {

      ifstream fp;
      nodo *nodotmp, *L = NULL;;
      volo tmp;
      char strOrario[6], strtmp[2+1];

      fp.open(nomeFile);
      if ( fp.fail() == true ) { cerr << "Il file non esiste! ";
                                 return L;
      }

      while ( fp.eof() == false ) {

         fp >> tmp.numero;
         fp >> tmp.dest;
         fp >> strOrario;
         strtmp[0]=strOrario[0]; strtmp[1]=strOrario[1]; strtmp[2]='\0';
         tmp.orario.hh = atoi(strtmp);
         strtmp[0]=strOrario[3]; strtmp[1]=strOrario[4]; strtmp[2]='\0';
         tmp.orario.mm = atoi(strtmp);
         fp >> tmp.postiLiberi;

         nodotmp = creaNodo(tmp);
         L = inserisciInTesta(L, nodotmp);
     }
     fp.close();
     return L;
}
```

```
void aggiornaPostiLiberiVolo(nodo* L, char numVolo[], int posti) {
     while ( L != NULL ) {
        if ( strcmp(L->info.numero, numVolo) == 0 ) {
          cout << "++++ Volo " << numVolo ;
          cout << " trovato e aggiornato ++++ \n\n";

          L->info.postiLiberi = posti;
          return;
         }
         L = L->next;
     }
     return;
}

nodo* eliminaVolo(nodo* L, char numVolo[]) {

     nodo* testaListaAggiornata = L;
     nodo* nodoPrima = NULL;
     while ( L != NULL ) {
          if ( strcmp(L->info.numero, numVolo) == 0 ) {
             cout << "++++ Volo " << numVolo ;
             cout << " trovato ed eliminato ++++ \n\n";
             if ( nodoPrima == NULL )
                testaListaAggiornata = L->next;
             else
                 nodoPrima->next = L->next;

             delete L;
             return testaListaAggiornata;
          }

          nodoPrima = L;
          L = L->next;
     }
     cout << " ++++ Volo non trovato ++++ \n\n ";
     return testaListaAggiornata;
}
```

ESERCIZIO 2

La Società Sportiva Olimpia dispone di un archivio in cui memorizza i farmaci che vengono assunti da ciascun atleta. L'archivio, realizzato attraverso un array dinamico in C++, consiste di un vettore di 22 componenti, ciascuna delle quali è costituita da una struttura in cui sono memorizzati:

- la matricola dell'atleta (un intero) ed
- il puntatore ad una lista dinamica semplicemente concatenata i cui dati sono costittuiti da:
  - la sigla del farmaco (una stringa di al più 12 caratteri);
  - la data della prescrizione della terapia.

La Commissione Nazionale Antidoping ha aggiornato l'elenco delle sostanze vietate. In particolare, ha vietato alcune sostanze che precedentemente erano ammesse, ed ha ammesso alcune sostanze che erano vietate. La Commissione ha prodotto e distribuito un file in cui sono memorizzate le nuove informazioni.
In particolare, ogni riga del file è costituito da tre campi, separati da uno spazio, rispettivamente:

- la sigla del farmaco (una striga di al più 12 caratteri);
- una cifra intera 0 o 1, ad indicare se il farmaco è vietato (cifra 0) o ammesso (cifra 1);
- la data a partire dalla quale il farmaco è vietato o ammesso, a seconda dei casi, rappresentata nel formato AAAA:MM:GG.

a) Definire i tipi di dato C++ adeguati a rappresentare ed elaborare le informazioni presentat, e una funzione di inizializzazione dell'archivio dei 22 atleti che restituisca un vettore dinamico con matricola nulla e lista farmaci vuota, per ogni cella del vettore riservata a un atleta .

b) Scrivere una funzione C++ che, assunti come parametri l'archivio degli atleti della Società Olimpia (come un vettore dinamico) e il nome del file contenente le nuove informazioni sui farmaci vietati o ammessi, elimini dall'archivio tutte le prescrizioni dei farmaci vietati avvenute NON ANTERIORMENTE alla data del divieto.

Esempio:

Si consideri in fig.(1) l'archivio di atleti ed in fig.(2) il file `NODOPING.DAT`.

In fig.(3) è mostrato l'archivio dopo l'esecuzione della funzione richiesta al punto (b)

Figura (1)

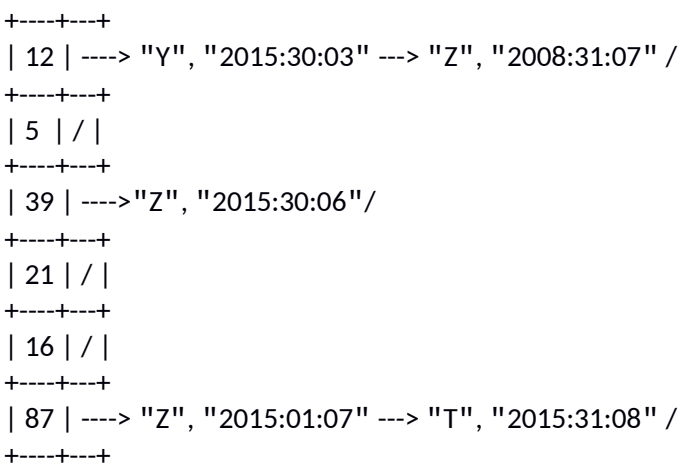
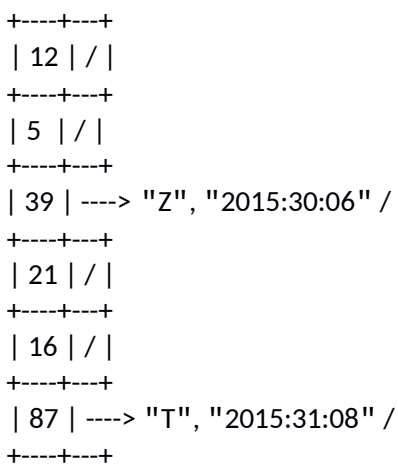


Figura (2)

```
Z 0 2015:01:07
X 1 2015:01:01
S 1 2015:30:06
Y 0 2014:01:01
R 1 2015:01:01
```

Figura (3)



```
#include <iostream>
#include <fstream>
#include <string.h>
#include <stdlib.h>

#define NUM_ATLETI 22
using namespace std;

struct nodo {
     char siglaFarmaco[12+1];
     char dataPrescrizioneTerapia[10+1];   // esempio: "2016:02:28"
     nodo* next;
};

struct atleta {
     int matricola;
     nodo* lf;     // testa della lista di farmaci
};

atleta* creaArchivioAtleti();

// Soluzione Domanda (b)
void aggiornaListeFarmaci(atleta archivio[], char nomeFile[]);
// aggiorna la lista di farmaci di ciascun atleta nell'archivio passato
// come parametro, controllando i divieti nel file, il cui nome e'
// passato come parametro.

// E' molto comodo scomporre il problema definendo altri
// due sottoprogrammi ausiliari.

listaFarmaci eliminaFarmaciVietati(listaFarmaci L, char nomeFile[]);
// aggiorna la lista dei farmaci passata come parametro, eliminando i
// nodi che si riferiscono ai divieti trovati nel file, il cui nome e'
// passato come parametro


int vietato(char* nomeFile, char* siglaFarmaco, char* dataInizioTerapia);
// questo sottoprogramma restituisce 1, se il farmaco la cui sigla e'
// passata come parametro, e' presente nel file come "vietato" con una
// data di inizio del divieto anteriore o uguale a quella passata come
// parametro.


int main() {

   // scrivere un programma principale per testare le funzioni
   // sviluppate, non e' richiesto dalla traccia dell'esercizio.

   return 0;
}
```

```
atleta* creaArchivioAtleti(){
       atleta* archivio = new atleta[NUM_ATLETI];
       for (int i = 0; i < NUM_ATLETI; ++i) {
          archivio.matricola = 0;
          archivio.lf = NULL;
       }
       return archivio;
}

void aggiornaListeFarmaci(atleta archivio[], char nomeFile[]) {
    for (int i = 0; i < NUM_ATLETI; ++i) {
       nodo* ptr = archivio[i].lf;
       ptr  = eliminaFarmaciVietati(ptr, nomeFile);
    }
}

nodo* eliminaFarmaciVietati(nodo* L, char nomeFile[]) {
   nodo* nuovaTesta = L, nodoPrima = NULL, *tmp;
   while ( L != NULL ) {

     if ( vietato(nomeFile, L->siglaFarmaco,
                            L-> dataPrescrizioneTerapia) == 1 ) {
       if ( nodoPrima == NULL )
         nuovaTesta = L->next;
       else
         nodoPrima->next = L->next;
       tmp = L;

     }
     else
       tmp = NULL;
     nodoPrima = L;
     L = L->next;
     if ( tmp != NULL ) delete tmp;

   }
   return nuovaTesta;
}

int vietato(char* nomeFile, char* siglaFarmaco, char* dataInizioTerapia){
    ifstream fp;
    fp.open(nomeFile, ios::in);
    if ( fp.fail() == true )
       return 0; // Fallimento all'apertura del file
    char buffSigla[12+1], buffData[10+1];
    int viet;
    while ( fp.eof() == false ) {
        fp >> buffSigla >> viet >> buffData;
        if ( viet == 1 && strcmp(buffData, dataInizioTerapia) <= 0 )
           return 1;
    }
    fp.close();
    return 0;
}
```