```
// Practica 1
#include <iostream>
#include <string.h>
#include <cstring>

using namespace std;

struct aeropuerto {
    char * nombre;
    struct vuelo * lista_de_vuelos;
    struct aeropuerto *link;
};

typedef aeropuerto Ngrafo;

struct vuelo {
    int id_vuelo;
    int fecha;
    struct aeropuerto *origen;
    struct aeropuerto *destino;
    struct vuelo * link;
};
typedef vuelo Narco;

void grafo_agregar_nodo (Ngrafo* &lista_n, char * nombre)
{
    Ngrafo* aux = lista_n;

    // Verificamos si que no exista ya el id_nodo.
    while (aux != NULL)
    {
        if (strcmp(aux->nombre,nombre)==0)
        {
            cout << "!!! Error: Ya existe un nodo con ese id_nodo." << endl;
            return;
        }
        aux = aux->link;
    }

    aux = new (Ngrafo);
    strcpy( aux->nombre,nombre );
    aux->lista_de_vuelos = NULL;
    aux->link = lista_n;
    lista_n = aux;
}
void grafo_agregar_arco (Ngrafo* lista_n, int id, char * id_nodo_origen, char *
id_nodo_destino)
{
    Ngrafo* nodo_origen = lista_n;
    Ngrafo* nodo_destino = lista_n;

    while (nodo_origen != NULL && nodo_origen->nombre != id_nodo_origen)
        nodo_origen = nodo_origen->link;

    while (nodo_destino != NULL && nodo_destino->nombre != id_nodo_destino)
        nodo_destino = nodo_destino->link;

    // Verificamos si existen id_nodo_origen y id_nodo_destino.
    if (nodo_origen == NULL || nodo_destino == NULL)
    {
        cout << "!!! Error: Alguno de los nodos no existe." << endl;
```

```
        return;
    }

    // Verificamos que no exista ya el id_arco.
    // Tarea :) Ponga aqu? su c?digo.

    Narco* aux = new (Narco);
    aux->id_vuelo = id;
    aux->destino = nodo_destino;
    aux->link= nodo_origen->lista_de_vuelos;
    nodo_origen->lista_de_vuelos = aux;

}
void grafo_mostrar (Ngrafo* lista_n)
{
    cout << "Grafo:\n\n";

    while (lista_n != NULL)
    {
        cout << "Nodo " << lista_n->nombre << ":" << endl;

        Narco* aux = lista_n->lista_de_vuelos;
        while (aux != NULL)
        {
            cout << "  Arco " << aux->id_vuelo << " -> Nodo " << aux->destino-
>nombre << endl;
            aux= aux->link;

        }

        lista_n = lista_n->link;
    }
    cout << endl;
    cout << endl;
}
void menu_opcion1 (Ngrafo* lista_n)
{
    grafo_mostrar (lista_n);
}
void menu_opcion2 (Ngrafo* &lista_n)
{
    char *id;

    cout << "Ingrese el id_nodo del nodo que desea incorporar: ";
    cin >> id;
    grafo_agregar_nodo (lista_n, id);
    cout << endl;
    cout << endl;
}
void menu_opcion3 (Ngrafo* &lista_n)
{
    int id;
    char *id_nodo_origen, *id_nodo_destino;

    cout << "Ingrese el id_arco del arco que desea incorporar: ";
    cin >> id;
    cout << "Ingrese el id_nodo del nodo desde donde sale el arco: ";
    cin >> id_nodo_origen;
    cout << "Ingrese el id_nodo del nodo a donde llega el arco: ";
    cin >> id_nodo_destino;
```

```
    grafo_agregar_arco (lista_n, id, id_nodo_origen, id_nodo_destino);

    cout << endl;
    cout << endl;
}

int main(int argc, char *argv[])
{
    Ngrafo* lista_n = NULL;
    bool c;
    int opcion = 0;
    do {
        cout << "************Menu de Opciones************\n";
        cout << endl;
        cout << "****** Grafos ******\n";
        cout << endl;
        cout << "1- Mostrar.\n";
        cout << "2- Insertar Nodo.\n";
        cout << "3- Insertar Arco.\n";
        cout << endl;
        cout << "    0- Salir\n";
        cout << endl;
        cout << "                          Ingrese opcion: ";
        cin >> opcion;
        cout << endl;
        cout << endl;

        switch(opcion)
        {
            case 1:
                menu_opcion1 (lista_n);
            break;
            case 2:
                menu_opcion2 (lista_n);
            break;
            case 3:
                menu_opcion3 (lista_n);
            break;

        }
    } while ( opcion != 0);
    return 0;
}

// Practica 2

#include <iostream>

using namespace std;

struct terminal {
    struct terminal* link;
    struct viaje* lista_de_viajes;
    int id;
    int cant_viajes;
    char ciudad[20];
};
typedef terminal Ngrafo;

struct viaje {
    int cod_viaje;
```

```
    int precio;
    struct terminal *destino;
    struct terminal *origen;
    struct viaje * link;
};
typedef viaje Narco;

void arco_nodo_baja (Ngrafo* &lista_n)
{
    int cod;
    cout<<"ingrese codigo"<<endl;
    cin>>cod;
    Ngrafo *aux =lista_n;
    Narco* act = lista_n->lista_de_viajes;
    Narco* ant = NULL;

  while (aux!=NULL){

    while(act!=NULL){
    if(act->cod_viaje==cod){
        break;
    }else{
    ant=act;
    act=act->link;
      }
}}
    if(act!=NULL&&ant==NULL){
       aux->lista_de_viajes=act->link;
        act=NULL;
        cout<<"se borro"<<endl;
    }
    else if (act!=NULL){
        ant->link=act->link;
        act=NULL;
       cout<<"se borro"<<endl;
    }

    aux=aux->link;

}

void grafo_agregar_nodo (Ngrafo* &lista_n, int id)
{
    Ngrafo* aux = lista_n;

    // Verificamos si que no exista ya el id_nodo.
    while (aux != NULL)
    {
        if (aux->id == id)
        {
            cout << "!!! Error: Ya existe un nodo con ese id_nodo." << endl;
            return;
        }
        aux = aux->link;
    }

    aux = new Ngrafo();
    aux->id = id;
    aux->lista_de_viajes = NULL;
    aux->link = lista_n;
    lista_n = aux;
}
```

```
void grafo_agregar_arco (Ngrafo* lista_n, int id, int id_nodo_origen, int
id_nodo_destino)
{
    Ngrafo* nodo_origen = lista_n;
    Ngrafo* nodo_destino = lista_n;

    while (nodo_origen != NULL && nodo_origen->id != id_nodo_origen)
        nodo_origen = nodo_origen->link;

    while (nodo_destino != NULL && nodo_destino->id != id_nodo_destino)
        nodo_destino = nodo_destino->link;

    // Verificamos si existen id_nodo_origen y id_nodo_destino.
    if (nodo_origen == NULL || nodo_destino == NULL)
    {
        cout << "!!! Error: Alguno de los nodos no existe." << endl;
        return;
    }

    // Verificamos que no exista ya el id_arco.
    // Tarea :) Ponga aqu? su c?digo.

    Narco* aux = new Narco();
    aux->cod_viaje = id;
    aux->destino = nodo_destino;
    aux->link = nodo_origen->lista_de_viajes;
    nodo_origen->lista_de_viajes = aux;
}

void grafo_mostrar (Ngrafo* lista_n)
{
    cout << "Grafo:\n\n";

    while (lista_n != NULL)
    {
        cout << "Nodo " << lista_n->id << ":" << endl;

        Narco* aux = lista_n->lista_de_viajes;
        while (aux != NULL)
        {
            cout << "  Arco " << aux->cod_viaje << " -> Nodo " << aux->destino-
>id << endl;
            aux= aux->link;
        }

        lista_n = lista_n->link;
    }
    cout << endl;
    cout << endl;
}

void menu_opcion1 (Ngrafo* lista_n)
{
    grafo_mostrar (lista_n);
}
void menu_opcion2 (Ngrafo* &lista_n)
{
    int id;

    cout << "Ingrese el id_nodo del nodo que desea incorporar: ";
    cin >> id;
```

```
    grafo_agregar_nodo (lista_n, id);
    cout << endl;
    cout << endl;
}
void menu_opcion3 (Ngrafo* &lista_n)
{
    int id, id_nodo_origen, id_nodo_destino;

    cout << "Ingrese el id_arco del arco que desea incorporar: ";
    cin >> id;
    cout << "Ingrese el id_nodo del nodo desde donde sale el arco: ";
    cin >> id_nodo_origen;
    cout << "Ingrese el id_nodo del nodo a donde llega el arco: ";
    cin >> id_nodo_destino;

    grafo_agregar_arco (lista_n, id, id_nodo_origen, id_nodo_destino);

    cout << endl;
    cout << endl;
}
void menu_opcion4 (Ngrafo* &lista_n){

    arco_nodo_baja (lista_n);

}

int main()
{

    Ngrafo* lista_n = NULL;
    bool c;
    int opcion = 0;
    do {
        cout << "************Menu de Opciones************\n";
        cout << endl;
        cout << "****** Grafos ******\n";
        cout << endl;
        cout << "1- Mostrar.\n";
        cout << "2- Insertar Nodo.\n";
        cout << "3- Insertar Arco.\n";
        cout << "4- Eliminar viaje.\n";
        //cout << "5- Contar cantidad de arcos y nodos.\n";
        cout << endl;
        cout << "    0- Salir\n";
        cout << endl;
        cout << "                         Ingrese opcion: ";
        cin >> opcion;
        cout << endl;
        cout << endl;

        switch(opcion)
        {
            case 1:
                menu_opcion1 (lista_n);
            break;
            case 2:
                menu_opcion2 (lista_n);
            break;
            case 3:
                menu_opcion3 (lista_n);
            break;
            case 4:
```

```
                menu_opcion4(lista_n);
            break;

        }
    } while ( opcion != 0);
return 0;
}
```

//Practica 2----

```
#include <iostream>
#include <iomanip>
#include <conio.h>
#include <stdlib.h>
using namespace std;

struct nodo_grafo{
    int id_nodo;
    struct nodo_arco* lista_arco;
    struct nodo_grafo* link;
};
typedef struct nodo_grafo NGrafo;

struct nodo_arco{
    int id_arco;
    NGrafo* destino;
    struct nodo_arco* link;
};
typedef struct nodo_arco NArco;

bool existeNodo(NGrafo* grafo, int _id_nodo);
void altaGrafoBack(NGrafo*& grafo, int _id_nodo);
void altaGrafoFront(NGrafo*& grafo, int _id_nodo);
void altaArco(NGrafo* &grafo,int _id_arco, int id_nodoOrigen, int
id_nodoDestino);
void mostrarRight(NGrafo* grafo,int _id_nodo);
void mostrarLeft(NGrafo* grafo, int id_nodo);
void eliminar_arco(NGrafo* grafo , int id_orig,int id_dest);

int main()
{
    NGrafo* grafo=NULL;
    altaGrafoFront(grafo, 12);
    altaGrafoFront(grafo, 21);
    altaGrafoFront(grafo, 33);
    altaGrafoFront(grafo, 54);
    altaGrafoFront(grafo, 765);
    //altaGrafoBack(grafo, 39);
    NGrafo* uno= grafo;
    NGrafo* dos= grafo;
    NArco* tres;
    while(uno!=NULL){
    cout<<uno->id_nodo<<"  ";
    uno= uno->link;
    }
    cout<<endl;

    altaArco(grafo, 112, 12, 21);
    altaArco(grafo, 223, 21, 33);
    altaArco(grafo, 114, 12, 54);
    altaArco(grafo, 227, 54, 33);
    altaArco(grafo, 116, 12, 765);
```

```
    //altaArco(grafo, 118, 12, 21);
    cout<<endl<<endl;

    /*while(dos!=NULL){
    cout<<"\nNODO:   "<<dos->id_nodo<<endl;
    tres= dos->lista_arco;
    cout<<endl;
        while(tres!=NULL){
             cout<<"---> "<<tres->destino->id_nodo;
            tres= tres->link;
        }

        dos= dos->link;
        cout<<endl;
    }

    cout<<"\n\nRight de 12:  "<<endl;
    mostrarRight(grafo, 12);*/
    cout<<"Elimino arco 112"<<endl;
    eliminar_arco(grafo,12,21);
    cout<<"Left  de 21:"<<endl;
    mostrarLeft(grafo,21);
    cout<<endl<<endl;

    return 0;
}

bool existeNodo(NGrafo* grafo, int _id_nodo){
    NGrafo* aux= grafo;
    bool band=true;
    if (aux==NULL){band=false;}

    else{
        while((aux!=NULL)&&(aux->id_nodo!= _id_nodo)){
            aux=aux->link;
        }
        if(aux==NULL){band=false;};
    }
    return band;
}

void altaGrafoBack(NGrafo*& grafo, int _id_nodo){
    if(!(existeNodo(grafo, _id_nodo))) return;

    NGrafo* aux= grafo;
    NGrafo* n_nodo= new NGrafo();
    n_nodo->id_nodo=_id_nodo;
    n_nodo->lista_arco=NULL;
    n_nodo->link=NULL;
    if(aux==NULL){grafo= n_nodo;}
    else{
        while(aux!=NULL){
            aux=aux->link;
        }
        aux->link= n_nodo;

    }
}

void altaGrafoFront(NGrafo*& grafo, int _id_nodo){
    bool band=existeNodo(grafo,_id_nodo);
```

```
 if(band==false){// controla si el dato ya existe antes que nada
    NGrafo* aux= grafo;
    grafo= new NGrafo();
    grafo->id_nodo= _id_nodo;
    grafo->lista_arco= NULL;
    grafo->link= aux;
  }
}

void altaArco(NGrafo* &grafo,int _id_arco, int id_nodoOrigen, int
id_nodoDestino){
    bool band1= false;
    bool band2= false;
    NGrafo* aux= grafo;
    NGrafo* aux2= NULL;
    NGrafo* aux3= NULL;
    NArco* n_arco;

    if(aux==NULL){cout<<"\nError, Grafo vacío"<<endl;
                  return;  }
    else{
        while ((aux!=NULL)&&(aux->id_nodo!=id_nodoOrigen)){
            aux=aux->link;
        }
        if(aux!=NULL){
                        band1=true;
                        aux2=aux; //nodo origen
                                                  }
        aux=grafo;
        while((aux!=NULL)&&(aux->id_nodo!=id_nodoDestino)){
            aux=aux->link;
        }
        if(aux!=NULL){
            band2=true;
            aux3=aux;//nodo destino
        }
        if((band1==true)&&(band2==true)){
            n_arco= new NArco();
            n_arco->id_arco=_id_arco;
            n_arco->destino=aux3 ;
            n_arco->link=aux2->lista_arco;
            aux2->lista_arco=n_arco;
        }
    }
}

void mostrarRight(NGrafo* grafo,int _id_nodo){

    NGrafo* aux= grafo;
    NArco* aux2;

    if(aux==NULL)return;
    else{
        while((aux!=NULL)&&(aux->id_nodo != _id_nodo)){
            aux= aux->link;
        }

        if(aux!=NULL){
            aux2= aux->lista_arco;}

            if(aux2==NULL)return;
```

```
        else{
            cout<<"\n\t";

            while (aux2 !=NULL){
              cout<<setw(4)<<aux2->destino->id_nodo;
                aux2= aux2->link;
            }
        }
    }
}

void mostrarLeft(NGrafo* grafo, int id_nodo){
    bool band=existeNodo(grafo, id_nodo);
    NGrafo *aux=grafo;
    NArco *arco=NULL;
    if(band==true){
        while (aux!=NULL) {
            if(aux->id_nodo!=id_nodo){
                arco=aux->lista_arco;
                while(arco!=NULL){
                    if(arco->destino->id_nodo==id_nodo){
                        cout<<aux->id_nodo<<endl;
                    }
                    arco=arco->link;
                }
            }
            aux=aux->link;
        }
    }
}
void eliminar_arco(NGrafo* grafo , int id_orig,int id_dest){
    NGrafo* aux=grafo;
    NArco* aux_a=NULL;
    NArco* aux_b=NULL;
    while((aux!=NULL)&&(aux->id_nodo!=id_orig)){
        aux=aux->link;
    }
    if(aux!=NULL){
        aux_a=aux->lista_arco;
        while((aux_a!=NULL)&&(aux_a->destino->id_nodo!=id_dest)){
            aux_b=aux_a;
            aux_a=aux_a->link;
        }
        if(aux_a!=NULL){
            aux_b=new NArco();
            aux_b->link=aux_a->link;
        }
        delete aux_a;
    }
    aux->lista_arco=aux_b;
}
```

//Practica de Matriz Adjunta

```
#include <iostream>
#define M 10
using namespace std;

void ej1RepMatAdjA(bool A[M][M]) {
//#define M 6

        for(int i=0;i<M;i++)
```

```
                for(int j=0;j<M;j++)
                        A[i][j]=false;

        A[0][1] = true;
        A[0][3] = true;
        A[1][2] = true;
        A[1][4] = true;
        A[2][4] = true;
        A[3][1] = true;
        A[4][3] = true;
        A[4][5] = true;
        A[5][2] = true;

        cout << "A|0 1 2 3 4 5 6 7 8 9  " << endl;
        cout << "--------------" << endl;
        for(int i=0;i<M;i++) {
                cout << i << "|";
                for(int j=0;j<M;j++) {
                        cout << A[i][j] << " ";
                }
                cout << endl;
        }
        cout << endl;
}

int ej2CantArcos(bool A[M][M]) {
        int cant = 0;
        for(int i=0;i<M;i++)
                for(int j=0;j<M;j++)
               if (A[i][j]) cant++;
        return cant;
}

void ej3Izquierdo(bool A[M][M],int nodo) {
        for(int i=0;i<M;i++)
                if (A[i][nodo]) cout << i << endl;
}

void ej4Adyacentes(bool A[M][M],int nodo) {
        for(int i=0;i<M;i++)
                if (A[nodo][i]) cout << i << endl;
}

void ej6Maximal(bool A[M][M]) {
        int cant;
        for(int i=0;i<M;i++) {
                cant = 0;
                for(int j=0;j<M;j++)
                  if (A[i][j]) cant++;
                if (cant==0) cout << i << endl;
        }
}

bool ej7minimo(bool A[M][M]) {
        int cant, k, minimal=0;
        for(int i=0;i<M;i++) {
                cant = 0;
                for(int j=0;j<M;j++)
                        if (A[j][i]) cant++;
                if (cant==0) {
                        minimal++;
                        k = i;
```

```
                }
        }
        if (minimal == 1) {
                cout << "minimo: " << k << endl;
                return true;
        } else cout << "sin minimo." << endl;

        return false;
}

int contarNodos(bool A[M][M]){
        bool vec[M];
        for(int i=0; i<M; i++){
                vec[i]=false;
        }
        int cont=0;

        for(int i=0; i<M; i++){
                for (int j=0; j<M; j++){
                        if(A[i][j]){
                                vec[i]=true;
                                vec[j]=true;

                        }
                }
        }
        for (int i=0; i<M; i++){
                if(vec[i]) cont++;
        }
        return cont;
}

int main(int argc, char *argv[]) {

          int nodo;
        // Ej 1:
        bool A[M][M];

        cout << "Ej 1: " << endl << endl;
        ej1RepMatAdjA(A);

        cout << "Ej 2: " << ej2CantArcos(A) << endl << endl;

        cout << "Ej 3 - nodos adyacentes: ";
        cin >> nodo;
        ej4Adyacentes(A,nodo);

        cout << endl;

        cout << "Ej 4 - Izquierdo de nodo: ";
        cin >> nodo;
        ej3Izquierdo(A,nodo);

        cout << "Ej 6 - Maximal: " << endl;
        ej6Maximal(A);
        cout << endl;

        cout << "Ej 7 - minimo: " << endl;
        ej7minimo(A);
        cout<<endl;
          cout<<" Cantidad de Nodos: " <<contarNodos(A)<<endl;

          return 0;
}
```