```
#include <iostream>
using namespace std;
struct listaDE{
  int dato;
  struct listaDE* anterior;
  struct listaDE* link;
};
typedef struct listaDE NlistaDE;
void pushBack(NlistaDE*& Nlista, int dat);
void pushOrden(NlistaDE*& Nlista, int dat);
bool lista_vacia(NlistaDE* Nlista);
void eliminar(NlistaDE* &NLista);
void draw(NlistaDE* Nlista);
void eliminarOcu(NlistaDE* &NLista, int dato);
int main(){
  NlistaDE* Nueva_lista = new NlistaDE;
  Nueva_lista = NULL;
}
void pushBack(NlistaDE* &Nlista, int dat){
  NlistaDE* auxiliar = Nlista;
  NlistaDE* nuevo_nodo = new(NlistaDE);
  nuevo_nodo->dato = dat;
  nuevo_nodo->link = NULL;
  if(lista_vacia(Nlista)){
     Nlista = nuevo_nodo;
     nuevo_nodo->anterior = NULL;
  }else{
     while(auxiliar->link!=NULL){
        auxiliar = auxiliar->link;
     }
     auxiliar->link = nuevo_nodo;
     nuevo_nodo->anterior = auxiliar;
  }
}
void pushOrden(NlistaDE*& Nlista, int dat){
  NlistaDE* auxiliar = Nlista;
  NlistaDE* nuevo_nodo = new(NlistaDE);
  nuevo_nodo->dato = dat;
  nuevo_nodo->link = NULL;
  nuevo_nodo->anterior = NULL;
  if(lista_vacia(Nlista)){
```

```
      Nlista = nuevo_nodo;
   }else{
      while(auxiliar->link!=NULL&&auxiliar->dato<dat){
         auxiliar = auxiliar->link;
      }
      if(auxiliar->link==NULL&&auxiliar->anterior==NULL){
         if(auxiliar->dato<dat){
            auxiliar->link = nuevo_nodo;
            nuevo_nodo->anterior = auxiliar;
         }else{
            nuevo_nodo->link= auxiliar;
            auxiliar->anterior = nuevo_nodo;
            Nlista = nuevo_nodo;
         }
      }else if(auxiliar->link==NULL){
         if(auxiliar->dato<dat){
            auxiliar->link = nuevo_nodo;
            nuevo_nodo->anterior = auxiliar;
         }else{
            nuevo_nodo->link= auxiliar;
            nuevo_nodo->anterior = auxiliar->anterior;
            auxiliar->anterior->link= nuevo_nodo;
            auxiliar->anterior = nuevo_nodo;
         }
      }else if(auxiliar->anterior==NULL){
         nuevo_nodo->link = Nlista;
         auxiliar->anterior = nuevo_nodo;
         Nlista = nuevo_nodo;
      }else{
         nuevo_nodo->link = auxiliar;
         nuevo_nodo->anterior = auxiliar->anterior;
         auxiliar->anterior->link = nuevo_nodo;
         auxiliar->anterior = nuevo_nodo;
      }
   }
}
bool lista_vacia(NlistaDE* Nlista){
   return Nlista == NULL;
}
void draw(NlistaDE* Nlista){
   if(lista_vacia(Nlista)){
```

```
        cout<<"Lista inexistente"<<endl;
    }else{
        while(Nlista!=NULL){
            cout<<"Dato: "<<Nlista->dato<<endl;
            Nlista = Nlista->link;
        }
    }
}
void eliminar(NlistaDE* &NLista){
    while(NLista!=NULL){
        NlistaDE* aux = NLista;
        NLista = NLista->link;
        delete aux;
    }
    cout<<"Eliminada"<<endl;
}
void eliminarOcu(NlistaDE* &NLista, int dato){
    NlistaDE* lista = NLista;
    NlistaDE* auxiliar = NULL;
    while(lista!=NULL&&lista->dato!=dato){
        lista = lista->link;
    }
    if(lista->anterior == NULL){
        auxiliar = lista;
        NLista = lista->link;
        lista->link->anterior = NULL;
        delete auxiliar;
    }else if(lista==NULL){
        std::cout<<"Dato no encontrado"<<std::endl;
    }else{
        if(lista->link==NULL){
            auxiliar = lista;
            lista->anterior->link = NULL;
            delete auxiliar;
        }else{
            auxiliar = lista;
            lista->anterior->link = lista->link;
            lista->link->anterior = lista->anterior;
            delete auxiliar;
        }
    }
```