```
#include <iostream>
using namespace std;
struct listaSE{
  int dato;
  struct listaSE* link;
};
typedef struct listaSE NlistaSE;
void pushBack(NlistaSE*& Nlista, int dat);
void pushBackOrden(NlistaSE*& Nlista, int dat);
void draw(NlistaSE* Nlista);
void inicia(NlistaSE* Nlista);
bool lista_vacia(NlistaSE* Nlista);
void eliminar(NlistaSE* &NLista);
void eliminarOcu(NlistaSE* &NLista, int dato);
void invertir(NlistaSE* &lista);
NlistaSE* combinar(NlistaSE* L1, NlistaSE* L2);
int main(){
  NlistaSE* NuevaLista = NULL;
  NlistaSE* NuevaLista2 = NULL;
  NlistaSE* nueva;
  pushBack(NuevaLista, 5);
  pushBack(NuevaLista, 4);
  pushBack(NuevaLista, 3);
  pushBack(NuevaLista2, 1);
  pushBack(NuevaLista2, 2);
  pushBack(NuevaLista2, 6);
  nueva = combinar(NuevaLista, NuevaLista2);
  draw(nueva);
  return 0;
}
void pushBack(NlistaSE* &Nlista, int dat){
  NlistaSE* auxiliar = Nlista;
  NlistaSE* nuevo_nodo = new(NlistaSE);
  nuevo_nodo->dato = dat;
  nuevo_nodo->link = NULL;
  if(lista_vacia(Nlista)){
     Nlista = nuevo_nodo;
  }else{
     while(auxiliar->link!=NULL){
        auxiliar = auxiliar->link;
     }
     auxiliar->link = nuevo_nodo;
  }
}
void draw(NlistaSE* Nlista){
  NlistaSE* aux = Nlista;
  while(aux!=NULL){
     cout<<"Dato: "<<aux->dato<<endl;
```

```
        aux = aux->link;
    }
}
bool lista_vacia(NlistaSE* Nlista){
    return Nlista==NULL;
}
void inicia(NlistaSE* Nlista){
    Nlista = NULL;
}
void pushBackOrden(NlistaSE*& Nlista, int dat){
    NlistaSE* ant = NULL;
    NlistaSE* auxiliar = Nlista;
    NlistaSE* nuevo_nodo = new(NlistaSE);
    nuevo_nodo->dato = dat;
    nuevo_nodo->link = NULL;
    if(lista_vacia(Nlista)){
        Nlista = nuevo_nodo;
    }else{
        while(auxiliar->link!=NULL&&auxiliar->dato<dat){
            ant = auxiliar;
            auxiliar = auxiliar->link;
        }
        if(auxiliar->link==NULL){
            auxiliar->link = nuevo_nodo;
        }else if(ant==NULL){
            nuevo_nodo->link = Nlista;
            Nlista = nuevo_nodo;
        }else{
            nuevo_nodo->link = ant->link;
            ant->link = nuevo_nodo;
        }
    }
}
void eliminar(NlistaSE* &NLista){
    while(NLista!=NULL){
        NlistaSE* aux = NLista;
        NLista = NLista->link;
        delete aux;
    }
    cout<<"Eliminada"<<endl;
}
void eliminarOcu(NlistaSE* &NLista, int dato){
    NlistaSE* actual = NLista;
    NlistaSE* anterior = NULL;
    while(actual!=NULL&&actual->dato!=dato){
        anterior = actual;
        actual = actual->link;
    }
```

```
  if(anterior==NULL){
     anterior = actual;
     NLista = actual->link;
     delete anterior;
  }else if(actual == NULL){
     cout<<"Dato no encontrado ELIMINACION"<<endl;
  }else{
     anterior->link = actual->link;
     delete actual;
  }
}
void invertir(NlistaSE* &lista){
  NlistaSE* actual = lista;
  NlistaSE* aux = lista;
  NlistaSE* anterior = NULL;
  while(aux!=NULL){
     aux = aux->link;
     actual->link = anterior;
     anterior = actual;
     actual = aux;
  }
  lista = anterior;
}
NlistaSE* combinar(NlistaSE* L1, NlistaSE* L2){
  NlistaSE* auxiliar = L1;
  NlistaSE* auxiliar2 = L2;
  NlistaSE* nuevaL3 = NULL;
  while(auxiliar!=NULL||auxiliar2!=NULL){
     if(auxiliar!=NULL){
        pushBack(nuevaL3, auxiliar->dato);
        auxiliar = auxiliar->link;
     }
     if(auxiliar2!=NULL){
        pushBack(nuevaL3,auxiliar2->dato);
        auxiliar2 = auxiliar2->link;
     }
  }
  NlistaSE* nuevaL = nuevaL3;
  return nuevaL;
}
```