LABORATORIO 7

ESTRUCTURA DE DATOS

MARIANA ESTEFANIA BARCENAS RODRIGUEZ
INGENIERIA EN COMPUTACION UAZ , 3°A

ACTIVIDAD 12:

código:

```
//estructura de datos
//mariana estefania barcenas rodriguez
// 01/03/2022

#include <conio.h>
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <stdbool.h>



typedef struct Nodos{
    int info;
    struct Nodos *liga;
}tiponodo;
typedef struct lista{
    struct Nodos *inicio;
    struct Nodos *fin;
    int t;
}tipolista;
typedef tiponodo *nodo;
typedef tipolista listaE;
listaE Lista;
nodo NuevoNodo(int,nodo);
listaE CrearFinal(listaE,int);//VA PA LA PILA
bool EsVacia(listaE);
void Recorrer(listaE);//VA PA LA PILA
listaE CrearInicio(listaE,int);
nodo BuscarX(listaE Lista,int,char);
listaE InsertarAntesX(listaE,int,int);
listaE InsertarDespuesX(listaE,int,int);
listaE EliminarX(listaE Lista,int ElemX);
int tsize(listaE);
listaE EliminarAntesX(listaE,int );
listaE EliminarDespuesX(listaE,int );
listaE EliminarInicio(listaE);
listaE EliminarFinal(listaE);//VA PA LA PILA
listaE OrdenarLista(listaE lista);
listaE InsertarAntesX(listaE lst,int Elem, int ElemX);
listaE ModificarElemX(listaE lst,int elementox,int elemento);
```

```
listaE EliminarDuplicados(listaE);


listaE CrearFinal(listaE lst,int Elem){
    if(EsVacia(lst)){
        lst.inicio=NuevoNodo(Elem,NULL);
        lst.fin=lst.inicio;
        lst.t++;
    }
    else{
        lst.fin->liga=NuevoNodo(Elem,NULL);
        lst.fin=lst.fin->liga;
        lst.t++;
    }
    Recorrer(lst);
    return lst;
 }
 nodo NuevoNodo(int elem,nodo liganueva){
    nodo n;
    n=(nodo)malloc(sizeof(tiponodo));
    if(!n){
        printf("Error al crear nodo nuevo");
        return NULL;
    }else{
        n->info=elem;
        n->liga=liganueva;
        return n;
    }
}

bool EsVacia(listaE lst){
    if(lst.inicio!=NULL)
        return false;
    else
        return true;
}

void Recorrer(listaE lst){
    nodo t;
    if(EsVacia(lst)){
        printf("La lista esta vacia\n");
    }
    else{
        t=lst.inicio;
        printf("\nEstado Actual del TDA Lista Enlazada Sencilla\n");
```

```
        while(t!=NULL){
            printf("%d | ",t->info);
            t=t->liga;
        }
    }
}

listaE CrearInicio(listaE lst,int Elem){
    nodo t;
    if(EsVacia(lst)){
        lst.inicio=NuevoNodo(Elem,NULL);
        lst.fin=lst.inicio;
        lst.t++;
    }
    else{
        t=NuevoNodo(Elem,NULL);
        t->liga=lst.inicio;
        lst.inicio=t;
        lst.t++;
    }
    Recorrer(lst);
    return lst;
}

nodo BuscarX(listaE lista,int ElemX,char Modo){
    nodo t=lista.inicio;
    nodo q=t;
    while(t!=NULL){
        if(t->info==ElemX){
            if(Modo=='A')
                return q;
            else
                return t;
        }
        else{
            q=t;
            t=t->liga;
        }
    }
    return NULL;
}

listaE InsertarAntesX(listaE lst,int Elem, int ElemX){
      nodo x;
      if(EsVacia(lst))
```

```
          printf("Lista Vacia");
      else{
          x=BuscarX(lst,ElemX,'A');
          if(x!=NULL){
              if(x==lst.inicio && x->info==ElemX)
                  lst=CrearInicio(lst,Elem);
              else{
                  x->liga=NuevoNodo(Elem,x->liga);
                  lst.t++;
                  Recorrer(lst);
              }
          }
          else{
              printf("\nElemento no encontrado\n");
              Recorrer(lst);
          }
      }
      return lst;
  }

listaE InsertarDespuesX(listaE lst,int Elem,int ElemX){
    nodo x;
    if(EsVacia(lst)){
        printf("Lista Vacia");
    }
    else{
        x=BuscarX(lst,ElemX,'D');
        if(x!=NULL){
            if(x->liga==NULL)
            {
                lst=CrearFinal(lst,Elem);
            }
            else{
                x->liga=NuevoNodo(Elem,x->liga);
                lst.t++;
                Recorrer(lst);
            }
        }else{
            printf("\nElemento no encontrado\n");
            Recorrer(lst);
        }
    }
    return lst;
}
```

```
listaE EliminarInicio(listaE lst){
    nodo aux;
    if (EsVacia(lst))
        printf("Es vacia");
    else{
        aux=lst.inicio;
        lst.inicio=aux->liga;
        lst.t--;
        free(aux);
    }
    Recorrer(lst);
    return lst;
}
listaE EliminarDespuesX(listaE lst,int ElemX){
    nodo x, aux;
    if (EsVacia(lst))
        printf("Es vacia");
    else{
        x=BuscarX(lst,ElemX,'D');
        if (x!=NULL){
            if (x->liga==NULL){
                printf("\n no se puede eliminar es fin de la estructura no
hay elemento despues de %d",ElemX);
            }
            else{
                aux=x->liga;
                x->liga=aux->liga;
                lst.t--;
                free(aux);
            }
        }
    }
    Recorrer(lst);
    return lst;
}
listaE EliminarFinal(listaE lst){
    nodo aux=lst.inicio;
    if (EsVacia(lst))
        printf("Es vacia");
    else{
        if (lst.inicio==lst.fin){
            lst.inicio=lst.fin=NULL;
            lst.t--;
            free(lst.inicio);
            free(lst.fin);
```

```
        }else{
            while(aux->liga!=lst.fin)
                aux=aux->liga;
            lst.fin=aux;
            lst.fin->liga=NULL;
            aux=aux->liga;
            lst.t--;
            free(aux);
        }
    }

    Recorrer(lst);
    return lst;
}

listaE EliminarX(listaE lst,int ElemX){
    nodo x,aux;
    if (EsVacia(lst))
        printf("Es vacia");
    else{
        x=BuscarX(lst,ElemX,'A');
        if (x!=NULL){
            if (x==lst.inicio && x->info==ElemX){
                lst=EliminarInicio(lst);
            }
            else if (x->liga==NULL){
                lst=EliminarFinal(lst);
            }
            else{
                aux=x->liga;
                x->liga=aux->liga;
                lst.t--;
                free(aux);
            }
        }
    }
    Recorrer(lst);
    return lst;
}

listaE EliminarAntesX(listaE lst,int ElemX){
    nodo x,aux;
    if (EsVacia(lst))
        printf("Es vacia");
    else{
```

```
        x=BuscarX(lst,ElemX,'D');
        if (x!=NULL){
            if(x==lst.inicio && x->info==ElemX)
                printf("No se elimino porque no hay elementos antes que
%d",ElemX);
            else{
                x=BuscarX(lst,ElemX,'A');
                if(x!=NULL){
                    if (x==lst.inicio){
                        lst=EliminarInicio(lst);}
                    else{
                        aux=BuscarX(lst,x->info,'A');
                        aux->liga=x->liga;
                        lst.t--;
                        free(x);
                    }
                }
            }
        }
    }
    Recorrer(lst);
    return lst;
}



int tsize(listaE lst){
    return lst.t;
}
listaE ModificarElemX(listaE lst,int elementox,int elemento){
    nodo n;
    n = (nodo)malloc(sizeof(tiponodo));
    n = lst.inicio;
    int encontrado = 0;
    if(lst.inicio!=NULL){
        while(n != NULL && encontrado != 1){
            if(n->info == elementox){
                n->info = elemento;
                printf("\nEl elemento ha sido modificado\n\n");
                encontrado = 1;
            }
            n = n->liga;
        }
        if(encontrado == 0){
            printf("\nElemento no encontrado\n\n");
```

```
        }
    }
    Recorrer(lst);
    return lst;
}

listaE OrdenarLista(listaE lst){
    nodo pivote = NULL,actual = NULL;
    int tmp;
    if (EsVacia(lst)){
        printf("\nLa lista esta vacia");
        return lst;
    }
    else{
        pivote = lst.inicio;
        while(pivote->liga != NULL ){
            actual = pivote->liga;
            while(actual != NULL){
                if(pivote->info > actual->info){
                    tmp = pivote->info;
                    pivote->info = actual->info;
                    actual->info = tmp;
                }
                actual = actual->liga;
            }
            pivote = pivote->liga;
        }
    }
    Recorrer(lst);
    return lst;
}

listaE EliminarDuplicados(listaE lst)
{
    nodo nx, aux,temp;
    if(EsVacia(lst))
    {
        printf("\n es vacia");
    }
    else
    {
        nx = lst.inicio;
        while (nx!=NULL)
        {
            aux =nx -> liga;
```

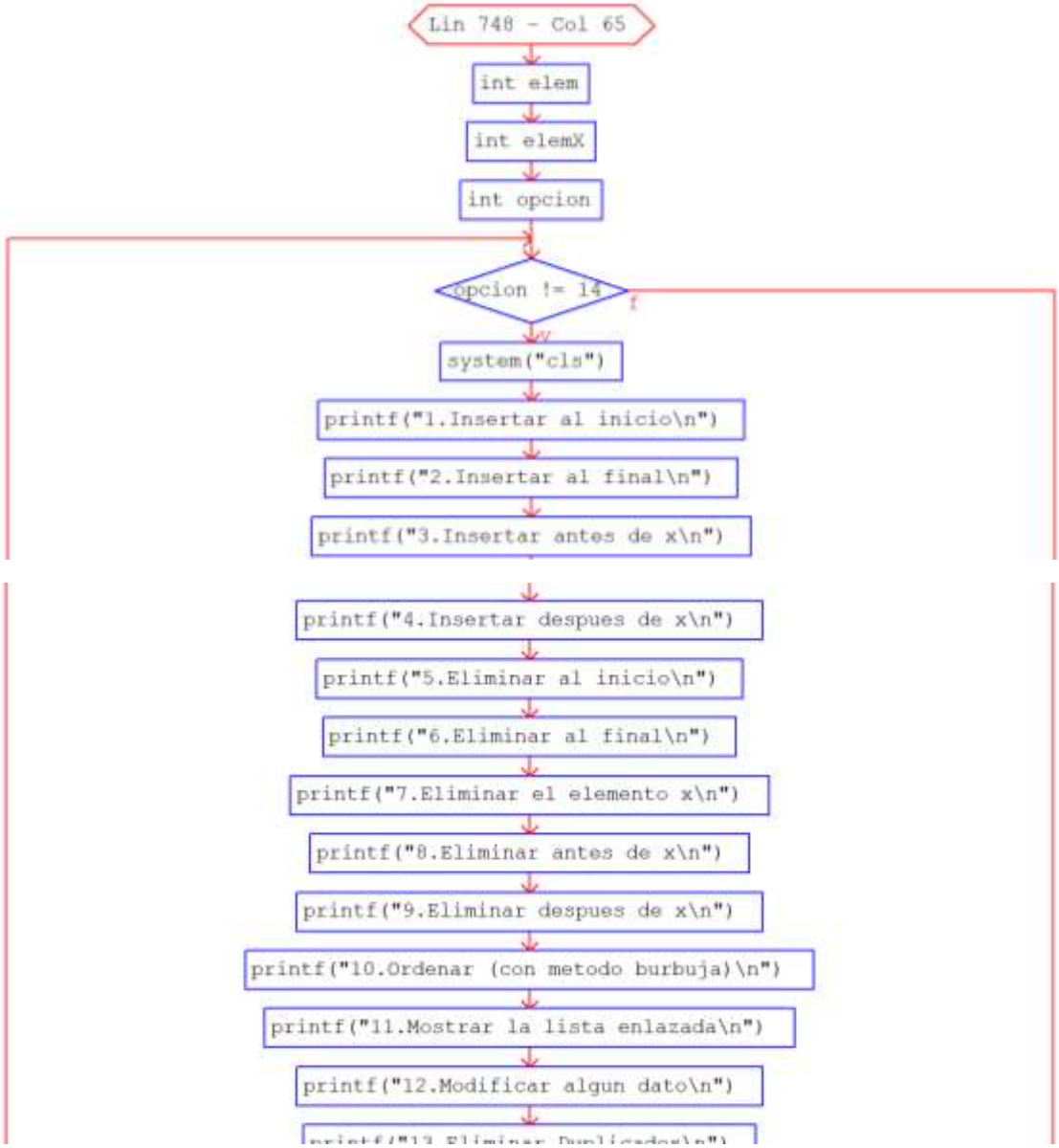
```
            temp = nx;
            while (aux!=NULL)
            {
                if (nx->info == aux->info)
                {
                    temp->liga = aux->liga;
                    if(aux==lst.fin)
                       lst.fin=temp;
                    free(aux);
                    aux=temp->liga;
                }
                else
                {
                    temp=aux;
                    aux = aux->liga;
                }

            }

            nx=nx->liga;
        }

    }
}



int main(int argc, char *argv[]) {
    int elem;
    int elemX;
    int opcion;
    while(opcion != 14){
        system("cls");
        printf("1.Insertar al inicio\n");
        printf("2.Insertar al final\n");
        printf("3.Insertar antes de x\n");
        printf("4.Insertar despues de x\n");
        printf("5.Eliminar al inicio\n");
        printf("6.Eliminar al final\n");
        printf("7.Eliminar el elemento x\n");
        printf("8.Eliminar antes de x\n");
        printf("9.Eliminar despues de x\n");
        printf("10.Ordenar (con metodo burbuja)\n");
        printf("11.Mostrar la lista enlazada\n");
        printf("12.Modificar algun dato\n");
```

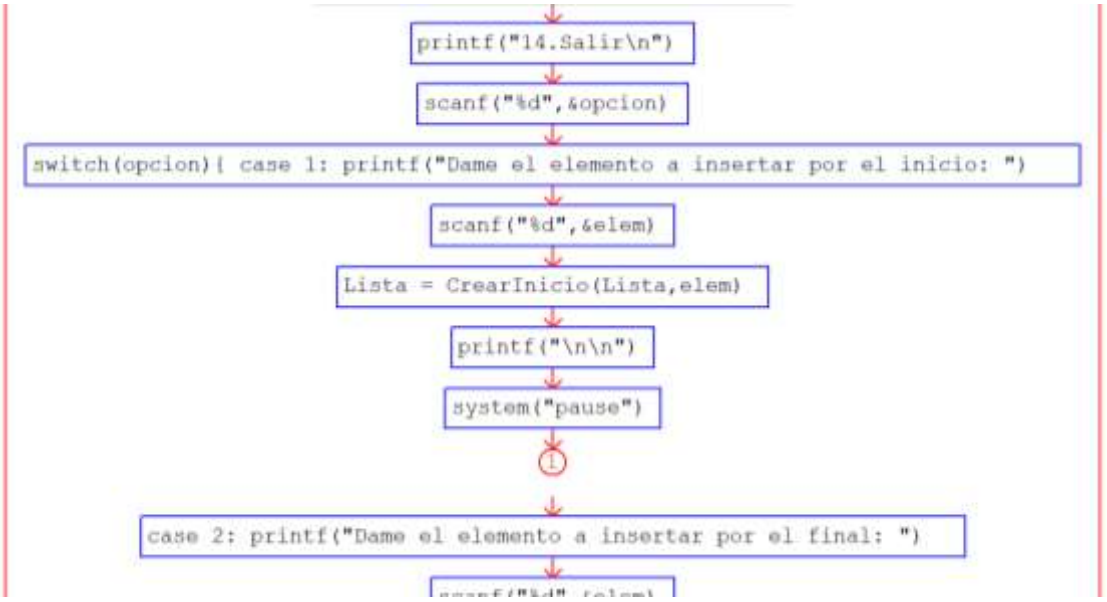
```
        printf("13.Eliminar Duplicados\n");
        printf("14.Salir\n");
        scanf("%d",&opcion);
        switch(opcion){
        case 1:
            printf("Dame el elemento a insertar por el inicio: ");
            scanf("%d",&elem);
            Lista = CrearInicio(Lista,elem);
            printf("\n\n");
            system("pause");
            break;
        case 2:
            printf("Dame el elemento a insertar por el final: ");
            scanf("%d",&elem);
            Lista = CrearFinal(Lista,elem);
            printf("\n\n");
            system("pause");
            break;
        case 3:
            printf("Ingrese el numero X: ");
            scanf("%d",&elemX);
            BuscarX(Lista,elemX,'A');
            printf("Dame el elemento a insertar antes de %d: ",elemX);
            scanf("%d",&elem);
            Lista=InsertarAntesX(Lista,elem,elemX);
            printf("\n\n");
            system("pause");
            break;
        case 4:
            printf("Ingrese el numero X: ");
            scanf("%d",&elemX);
            BuscarX(Lista,elemX,'D');
            printf("Dame el elemento a insertar despues de %d: ",elemX);
            scanf("%d",&elem);
            Lista=InsertarDespuesX(Lista,elem,elemX);
            printf("\n\n");
            system("pause");
            break;
        case 5:
            Lista=EliminarInicio(Lista);
            printf("\n\n");
            system("pause");
            break;
        case 6:
            Lista=EliminarFinal(Lista);
```

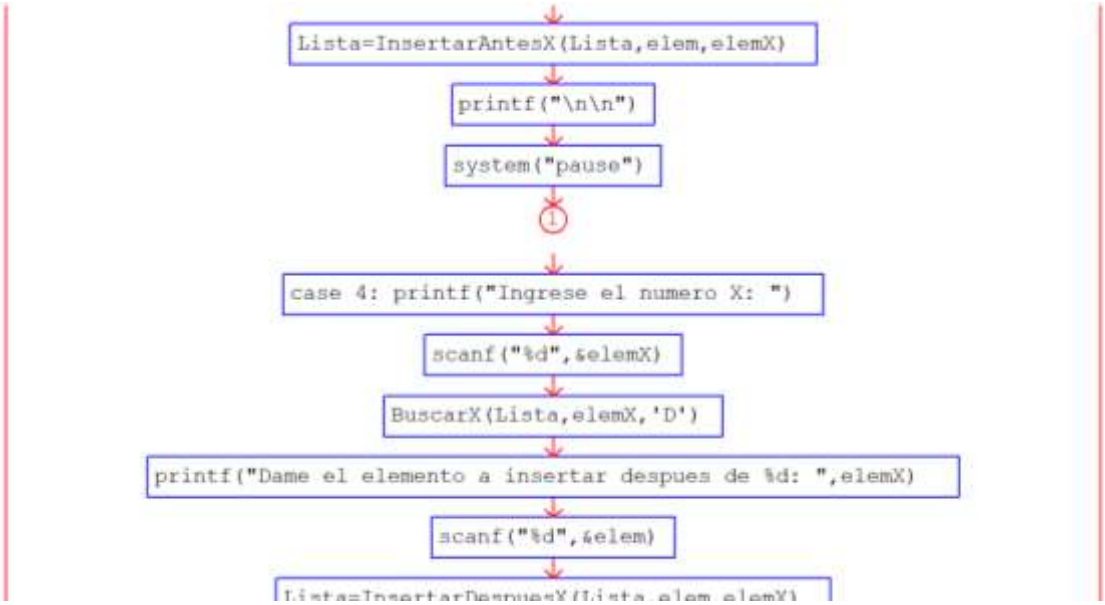
```
            printf("\n\n");
            system("pause");
            break;
        case 7:
            printf("Ingrese el numero X: ");
            scanf("%d",&elemX);
            Lista=EliminarX(Lista,elemX);
            printf("\n\n");
            system("pause");
            break;
        case 8:
            printf("Ingrese el numero X: ");
            scanf("%d",&elemX);
            BuscarX(Lista,elemX,'A');
            Lista=EliminarAntesX(Lista,elemX);
            printf("\n\n");
            system("pause");
            break;
        case 9:
            printf("Ingrese el numero X: ");
            scanf("%d",&elemX);
            BuscarX(Lista,elemX,'D');
            Lista=EliminarDespuesX(Lista,elemX);
            printf("\n\n");
            system("pause");
            break;
        case 10:
            Lista=OrdenarLista(Lista);
            printf("\n\n");
            system("pause");
            break;
        case 11:
            Recorrer(Lista);
            printf("\n\n");
            system("pause");
            break;
        case 12:
            printf("Ingrese el numero X: ");
            scanf("%d",&elemX);
            printf("Dame el nuevo elemento que lo va a sustituir: ");
            scanf("%d",&elem);
            Lista = ModificarElemX(Lista,elemX,elem);
            printf("\n\n");
            system("pause");
            break;
```

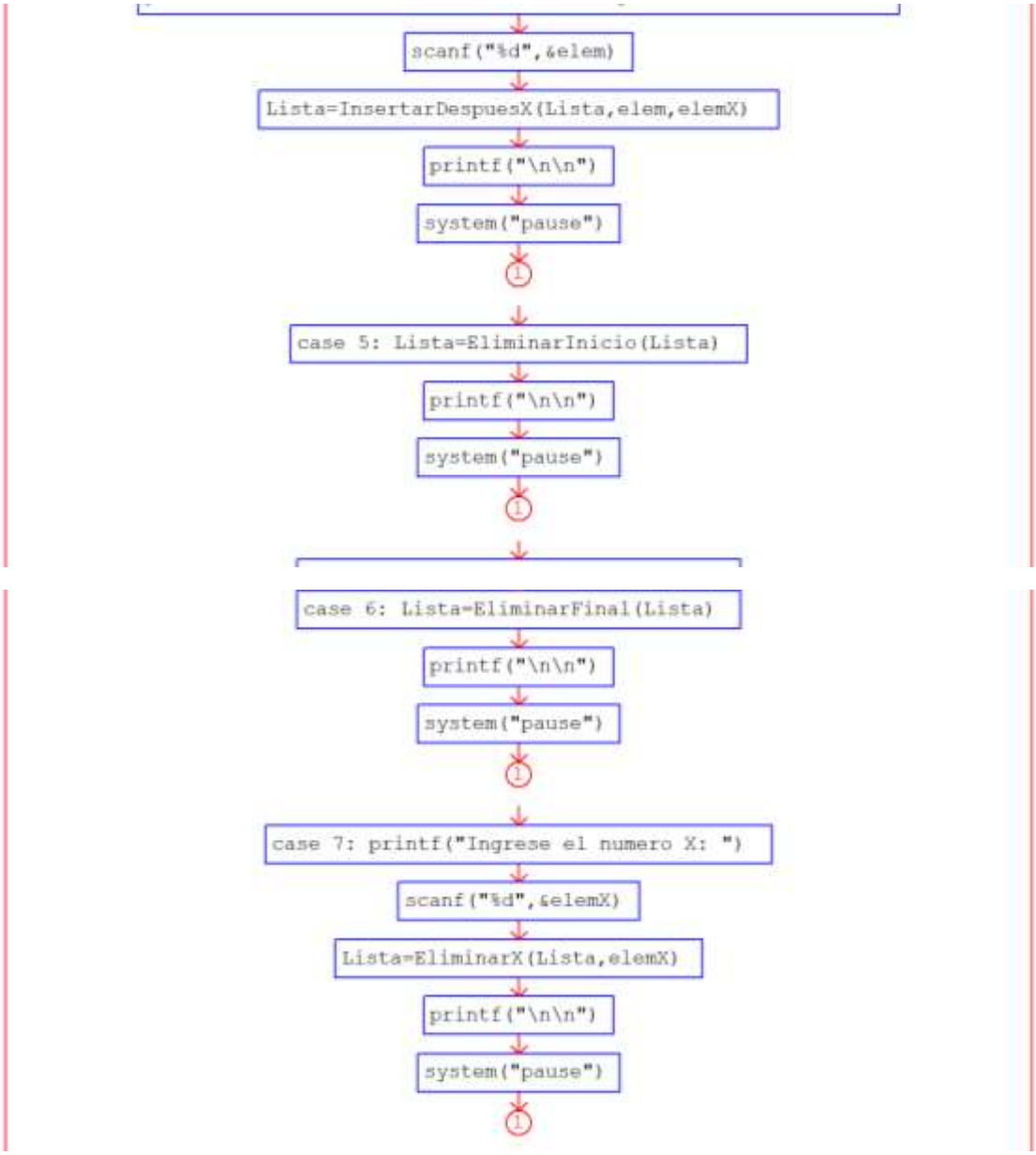
```
        case 13:
           printf("Elimina elementos duplicados");
           Lista=EliminarDuplicados(Lista);
           Recorrer(Lista);
           printf("\n\n");
           system("pause");
            break;
        case 14:
            break;
        }
    }
    printf("\nHasta luego\n");
    system("pause");
    return 0;
}
```

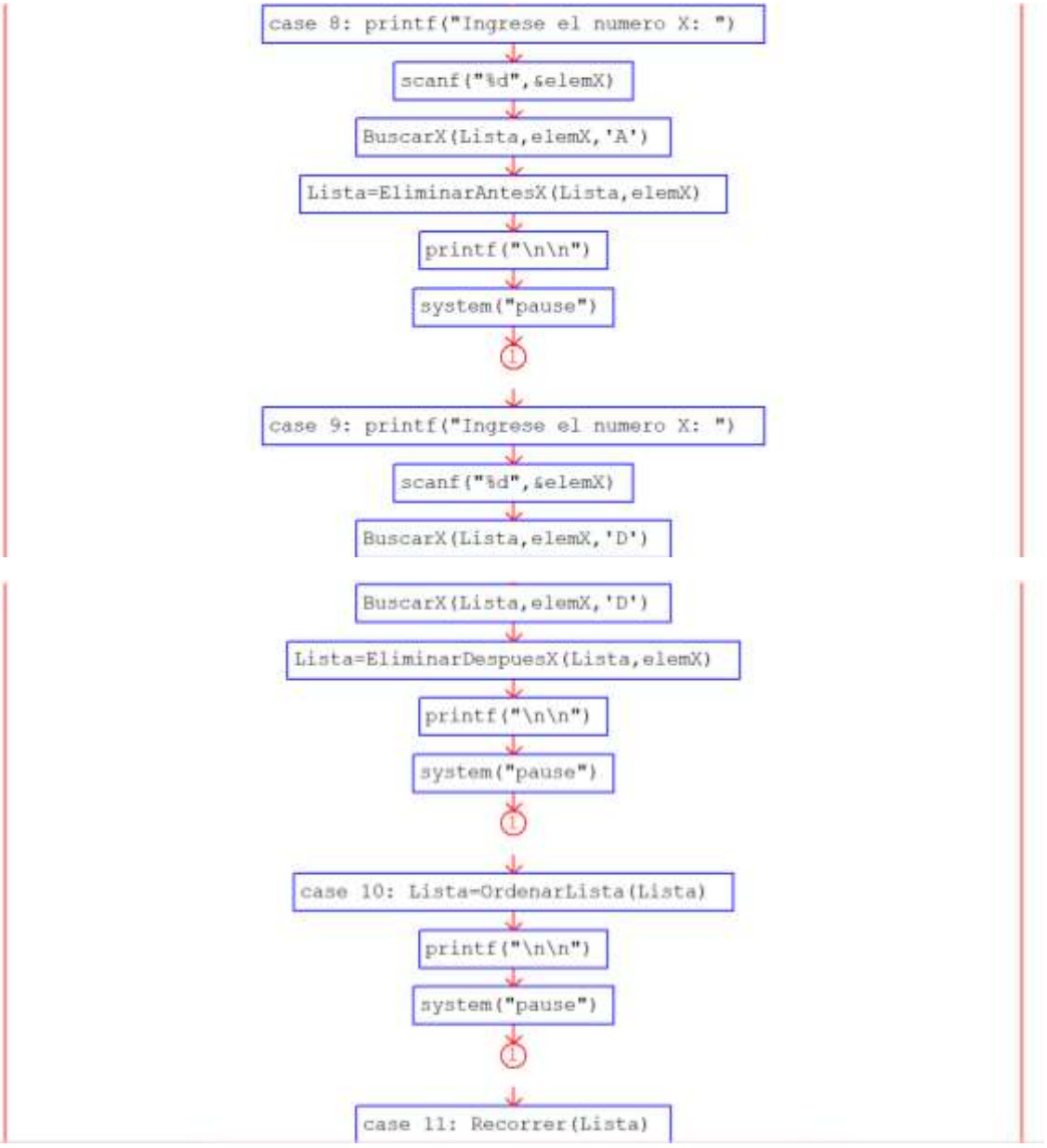
diagrama:

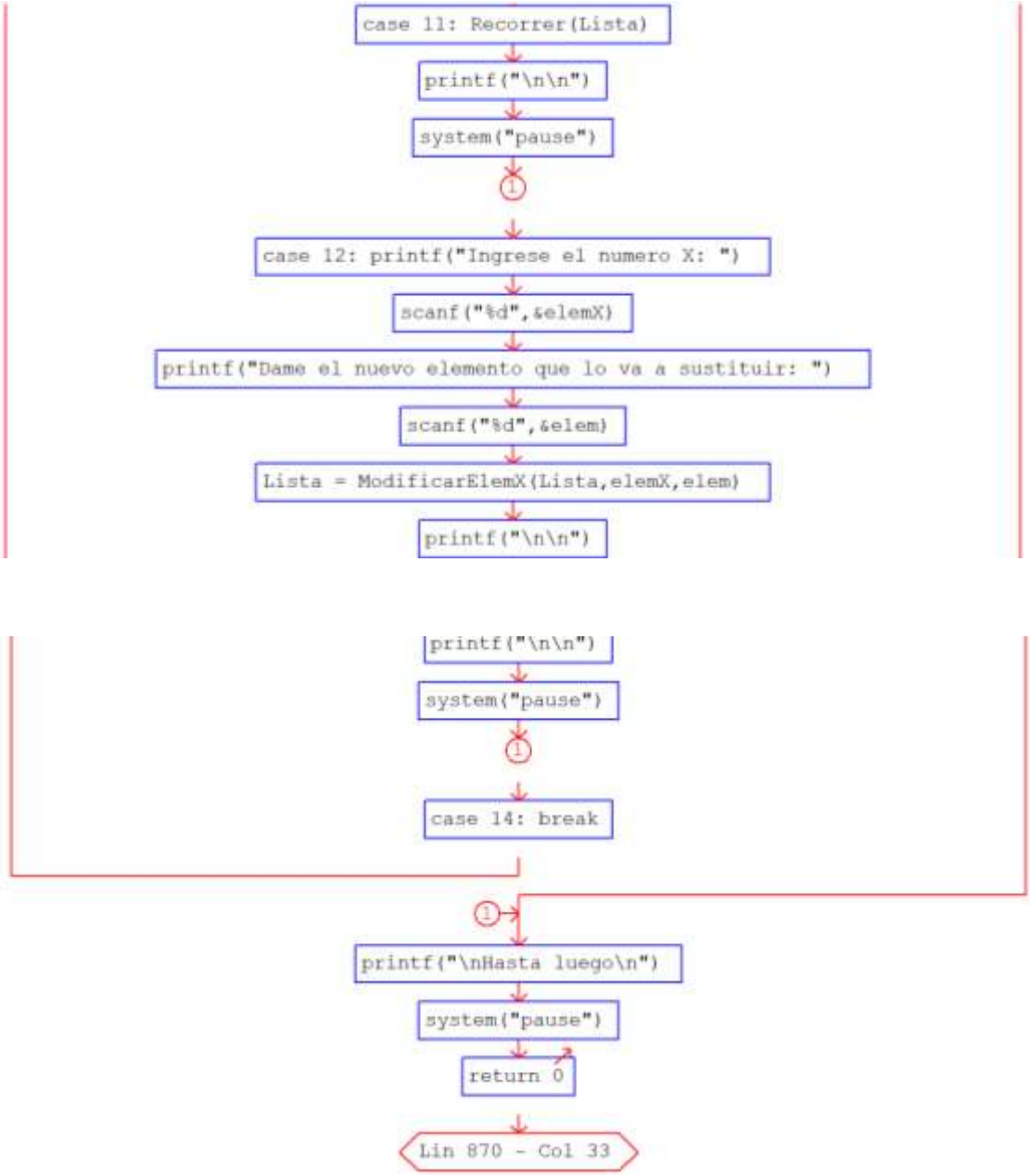

Lin 748 - Col 65
int elem
int elemX
int opcion
opcion != 14
f
v
system("cls")
printf("1.Insertar al inicio\n")
printf("2.Insertar al final\n")
printf("3.Insertar antes de x\n")
printf("4.Insertar despues de x\n")
printf("5.Eliminar al inicio\n")
printf("6.Eliminar al final\n")
printf("7.Eliminar el elemento x\n")
printf("8.Eliminar antes de x\n")
printf("9.Eliminar despues de x\n")
printf("10.Ordenar (con metodo burbuja)\n")
printf("11.Mostrar la lista enlazada\n")
printf("12.Modificar algun dato\n")


printf("14.Salir\n")
scanf("%d",&opcion)
switch(opcion){ case 1: printf("Dame el elemento a insertar por el inicio: ")
scanf("%d",&elem)
Lista = CrearInicio(Lista,elem)
printf("\n\n")
system("pause")
1
case 2: printf("Dame el elemento a insertar por el final: ")


Lista=InsertarAntesX(Lista,elem,elemX)
printf("\n\n")
system("pause")
1
case 4: printf("Ingrese el numero X: ")
scanf("%d",&elemX)
BuscarX(Lista,elemX,'D')
printf("Dame el elemento a insertar despues de %d: ",elemX)
scanf("%d",&elem)


scanf("%d",&elem)
Lista=InsertarDespuesX(Lista,elem,elemX)
printf("\n\n")
system("pause")
1
case 5: Lista=EliminarInicio(Lista)
printf("\n\n")
system("pause")
1
case 6: Lista=EliminarFinal(Lista)
printf("\n\n")
system("pause")
1
case 7: printf("Ingrese el numero X: ")
scanf("%d",&elemX)
Lista=EliminarX(Lista,elemX)
printf("\n\n")
system("pause")
1


case 8: printf("Ingrese el numero X: ")
scanf("%d",&elemX)
BuscarX(Lista,elemX,'A')
Lista=EliminarAntesX(Lista,elemX)
printf("\n\n")
system("pause")
1
case 9: printf("Ingrese el numero X: ")
scanf("%d",&elemX)
BuscarX(Lista,elemX,'D')
BuscarX(Lista,elemX,'D')
Lista=EliminarDespuesX(Lista,elemX)
printf("\n\n")
system("pause")
1
case 10: Lista=OrdenarLista(Lista)
printf("\n\n")
system("pause")
1
case 11: Recorrer(Lista)



ACTIVIDAD 13:

código:

```
//estructura de datos
//mariana estefania barcenas rodriguez
// 01/03/2022

#include <conio.h>
#include <stdio.h>
#include <malloc.h>
```

```c
#include <stdlib.h>
#include <stdbool.h>



typedef struct Nodos{
    int info;
    struct Nodos *liga;
}tiponodo;
typedef struct lista{
    struct Nodos *inicio;
    struct Nodos *fin;
    int t;
}tipolista;
typedef tiponodo *nodo;
typedef tipolista listaE;
listaE Lista;
nodo NuevoNodo(int,nodo);
listaE CrearFinal(listaE,int);//VA PA LA PILA
bool EsVacia(listaE);
void Recorrer(listaE);//VA PA LA PILA
listaE CrearInicio(listaE,int);
nodo BuscarX(listaE Lista,int,char);
listaE InsertarAntesX(listaE,int,int);
listaE InsertarDespuesX(listaE,int,int);
listaE EliminarX(listaE Lista,int ElemX);
int tsize(listaE);
listaE EliminarAntesX(listaE,int );
listaE EliminarDespuesX(listaE,int );
listaE EliminarInicio(listaE);
listaE EliminarFinal(listaE);//VA PA LA PILA
listaE OrdenarLista(listaE lista);
listaE InsertarAntesX(listaE lst,int Elem, int ElemX);
listaE ModificarElemX(listaE lst,int elementox,int elemento);
listaE EliminarDuplicados(listaE);


listaE CrearFinal(listaE lst,int Elem){
    if(EsVacia(lst)){
        lst.inicio=NuevoNodo(Elem,NULL);
        lst.fin=lst.inicio;
        lst.t++;
    }
    else{
        lst.fin->liga=NuevoNodo(Elem,NULL);
```

```
        lst.fin=lst.fin->liga;
        lst.t++;
    }
    Recorrer(lst);
    return lst;
 }
 nodo NuevoNodo(int elem,nodo liganueva){
    nodo n;
    n=(nodo)malloc(sizeof(tiponodo));
    if(!n){
        printf("Error al crear nodo nuevo");
        return NULL;
    }else{
        n->info=elem;
        n->liga=liganueva;
        return n;
    }
}

bool EsVacia(listaE lst){
    if(lst.inicio!=NULL)
        return false;
    else
        return true;
}

void Recorrer(listaE lst){
    nodo t;
    if(EsVacia(lst)){
        printf("La lista esta vacia\n");
    }
    else{
        t=lst.inicio;
        printf("\nEstado Actual del TDA Lista Enlazada Sencilla\n");
        while(t!=NULL){
            printf("%d | ",t->info);
            t=t->liga;
        }
    }
}

listaE CrearInicio(listaE lst,int Elem){
    nodo t;
    if(EsVacia(lst)){
        lst.inicio=NuevoNodo(Elem,NULL);
```

```
        lst.fin=lst.inicio;
        lst.t++;
    }
    else{
        t=NuevoNodo(Elem,NULL);
        t->liga=lst.inicio;
        lst.inicio=t;
        lst.t++;
    }
    Recorrer(lst);
    return lst;
}

nodo BuscarX(listaE lista,int ElemX,char Modo){
    nodo t=lista.inicio;
    nodo q=t;
    while(t!=NULL){
        if(t->info==ElemX){
            if(Modo=='A')
                return q;
            else
                return t;
        }
        else{
            q=t;
            t=t->liga;
        }
    }
    return NULL;
}

listaE InsertarAntesX(listaE lst,int Elem, int ElemX){
      nodo x;
      if(EsVacia(lst))
          printf("Lista Vacia");
      else{
          x=BuscarX(lst,ElemX,'A');
          if(x!=NULL){
              if(x==lst.inicio && x->info==ElemX)
                  lst=CrearInicio(lst,Elem);
              else{
                  x->liga=NuevoNodo(Elem,x->liga);
                  lst.t++;
                  Recorrer(lst);
              }
```

```
          }
          else{
              printf("\nElemento no encontrado\n");
              Recorrer(lst);
          }
      }
      return lst;
  }

listaE InsertarDespuesX(listaE lst,int Elem,int ElemX){
    nodo x;
    if(EsVacia(lst)){
        printf("Lista Vacia");
    }
    else{
        x=BuscarX(lst,ElemX,'D');
        if(x!=NULL){
            if(x->liga==NULL)
            {
                lst=CrearFinal(lst,Elem);
            }
            else{
                x->liga=NuevoNodo(Elem,x->liga);
                lst.t++;
                Recorrer(lst);
            }
        }else{
            printf("\nElemento no encontrado\n");
            Recorrer(lst);
        }
    }
    return lst;
}

listaE EliminarInicio(listaE lst){
    nodo aux;
    if (EsVacia(lst))
        printf("Es vacia");
    else{
        aux=lst.inicio;
        lst.inicio=aux->liga;
        lst.t--;
        free(aux);
    }
    Recorrer(lst);
```

```
    return lst;
}
listaE EliminarDespuesX(listaE lst,int ElemX){
    nodo x, aux;
    if (EsVacia(lst))
        printf("Es vacia");
    else{
        x=BuscarX(lst,ElemX,'D');
        if (x!=NULL){
            if (x->liga==NULL){
                printf("\n no se puede eliminar es fin de la estructura no
hay elemento despues de %d",ElemX);
            }
            else{
                aux=x->liga;
                x->liga=aux->liga;
                lst.t--;
                free(aux);
            }
        }
    }
    Recorrer(lst);
    return lst;
}
listaE EliminarFinal(listaE lst){
    nodo aux=lst.inicio;
    if (EsVacia(lst))
        printf("Es vacia");
    else{
        if (lst.inicio==lst.fin){
            lst.inicio=lst.fin=NULL;
            lst.t--;
            free(lst.inicio);
            free(lst.fin);
        }else{
            while(aux->liga!=lst.fin)
                aux=aux->liga;
            lst.fin=aux;
            lst.fin->liga=NULL;
            aux=aux->liga;
            lst.t--;
            free(aux);
        }
    }
```

```
    Recorrer(lst);
    return lst;
}

listaE EliminarX(listaE lst,int ElemX){
    nodo x,aux;
    if (EsVacia(lst))
        printf("Es vacia");
    else{
        x=BuscarX(lst,ElemX,'A');
        if (x!=NULL){
            if (x==lst.inicio && x->info==ElemX){
                lst=EliminarInicio(lst);
            }
            else if (x->liga==NULL){
                lst=EliminarFinal(lst);
            }
            else{
                aux=x->liga;
                x->liga=aux->liga;
                lst.t--;
                free(aux);
            }
        }
    }
    Recorrer(lst);
    return lst;
}

listaE EliminarAntesX(listaE lst,int ElemX){
    nodo x,aux;
    if (EsVacia(lst))
        printf("Es vacia");
    else{
        x=BuscarX(lst,ElemX,'D');
        if (x!=NULL){
            if(x==lst.inicio && x->info==ElemX)
                printf("No se elimino porque no hay elementos antes que
%d",ElemX);
            else{
                x=BuscarX(lst,ElemX,'A');
                if(x!=NULL){
                    if (x==lst.inicio){
                        lst=EliminarInicio(lst);}
                    else{
```

```
                        aux=BuscarX(lst,x->info,'A');
                        aux->liga=x->liga;
                        lst.t--;
                        free(x);
                    }
                }
            }
        }
    }
    Recorrer(lst);
    return lst;
}



int tsize(listaE lst){
    return lst.t;
}
listaE ModificarElemX(listaE lst,int elementox,int elemento){
    nodo n;
    n = (nodo)malloc(sizeof(tiponodo));
    n = lst.inicio;
    int encontrado = 0;
    if(lst.inicio!=NULL){
        while(n != NULL && encontrado != 1){
            if(n->info == elementox){
                n->info = elemento;
                printf("\nEl elemento ha sido modificado\n\n");
                encontrado = 1;
            }
            n = n->liga;
        }
        if(encontrado == 0){
            printf("\nElemento no encontrado\n\n");
        }
    }
    Recorrer(lst);
    return lst;
}

listaE OrdenarLista(listaE lst){
    nodo pivote = NULL,actual = NULL;
    int tmp;
    if (EsVacia(lst)){
        printf("\nLa lista esta vacia");
```

```
        return lst;
    }
    else{
        pivote = lst.inicio;
        while(pivote->liga != NULL ){
            actual = pivote->liga;
            while(actual != NULL){
                if(pivote->info > actual->info){
                    tmp = pivote->info;
                    pivote->info = actual->info;
                    actual->info = tmp;
                }
                actual = actual->liga;
            }
            pivote = pivote->liga;
        }
    }
    Recorrer(lst);
    return lst;
}

listaE EliminarDuplicados(listaE lst)
{
    nodo nx, aux,temp;
    if(EsVacia(lst))
    {
        printf("\n es vacia");
    }
    else
    {
        nx = lst.inicio;
        while (nx!=NULL)
        {
            aux =nx -> liga;
            temp = nx;
            while (aux!=NULL)
            {
                if (nx->info == aux->info)
                {
                    temp->liga = aux->liga;
                    if(aux==lst.fin)
                       lst.fin=temp;
                    free(aux);
                    aux=temp->liga;
                }
```

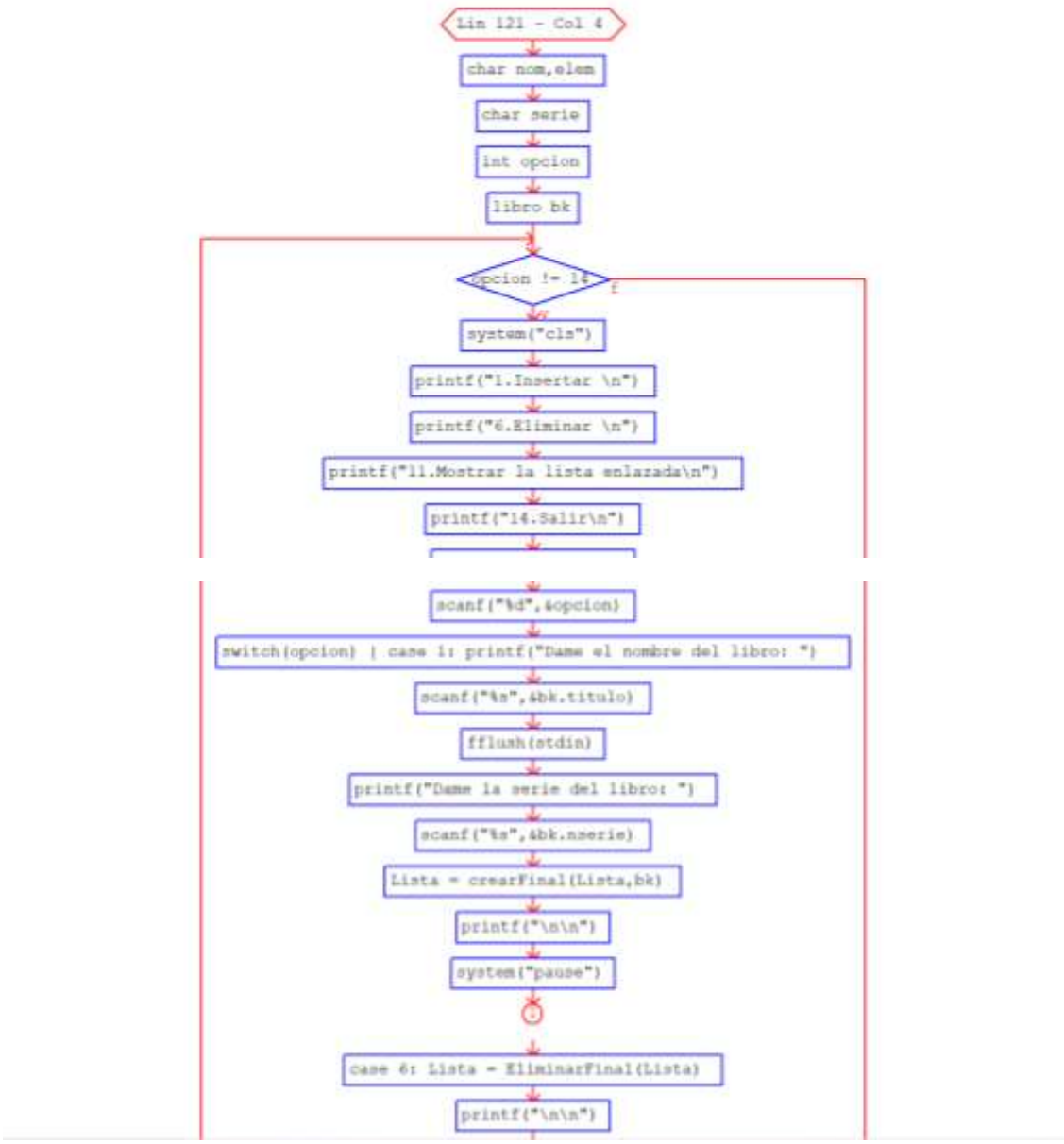
```
                else
                {
                    temp=aux;
                    aux = aux->liga;
                }

            }

            nx=nx->liga;
        }

    }
}



int main(int argc, char *argv[]) {
    int elem;
    int elemX;
    int opcion;
    while(opcion != 14){
        system("cls");
        printf("1.Insertar al inicio\n");
        printf("2.Insertar al final\n");
        printf("3.Insertar antes de x\n");
        printf("4.Insertar despues de x\n");
        printf("5.Eliminar al inicio\n");
        printf("6.Eliminar al final\n");
        printf("7.Eliminar el elemento x\n");
        printf("8.Eliminar antes de x\n");
        printf("9.Eliminar despues de x\n");
        printf("10.Ordenar (con metodo burbuja)\n");
        printf("11.Mostrar la lista enlazada\n");
        printf("12.Modificar algun dato\n");
        printf("13.Eliminar Duplicados\n");
        printf("14.Salir\n");
        scanf("%d",&opcion);
        switch(opcion){
        case 1:
            printf("Dame el elemento a insertar por el inicio: ");
            scanf("%d",&elem);
            Lista = CrearInicio(Lista,elem);
            printf("\n\n");
            system("pause");
            break;
```

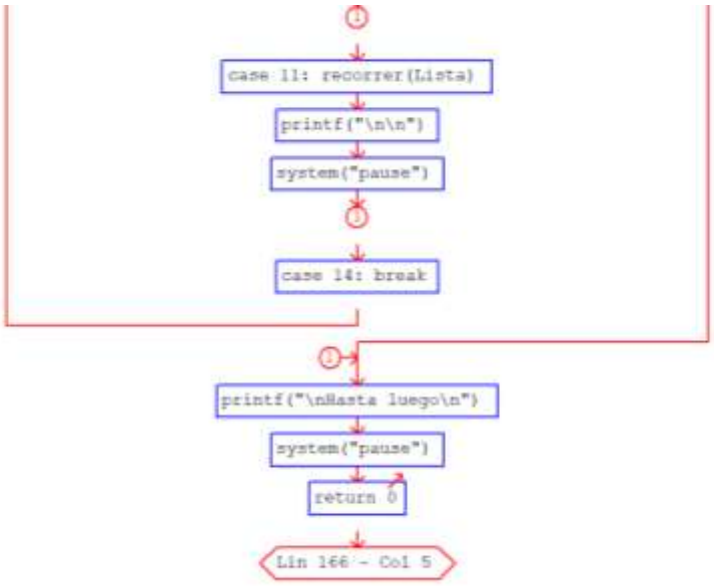
```
        case 2:
            printf("Dame el elemento a insertar por el final: ");
            scanf("%d",&elem);
            Lista = CrearFinal(Lista,elem);
            printf("\n\n");
            system("pause");
            break;
        case 3:
            printf("Ingrese el numero X: ");
            scanf("%d",&elemX);
            BuscarX(Lista,elemX,'A');
            printf("Dame el elemento a insertar antes de %d: ",elemX);
            scanf("%d",&elem);
            Lista=InsertarAntesX(Lista,elem,elemX);
            printf("\n\n");
            system("pause");
            break;
        case 4:
            printf("Ingrese el numero X: ");
            scanf("%d",&elemX);
            BuscarX(Lista,elemX,'D');
            printf("Dame el elemento a insertar despues de %d: ",elemX);
            scanf("%d",&elem);
            Lista=InsertarDespuesX(Lista,elem,elemX);
            printf("\n\n");
            system("pause");
            break;
        case 5:
            Lista=EliminarInicio(Lista);
            printf("\n\n");
            system("pause");
            break;
        case 6:
            Lista=EliminarFinal(Lista);
            printf("\n\n");
            system("pause");
            break;
        case 7:
            printf("Ingrese el numero X: ");
            scanf("%d",&elemX);
            Lista=EliminarX(Lista,elemX);
            printf("\n\n");
            system("pause");
            break;
        case 8:
```

```
            printf("Ingrese el numero X: ");
            scanf("%d",&elemX);
            BuscarX(Lista,elemX,'A');
            Lista=EliminarAntesX(Lista,elemX);
            printf("\n\n");
            system("pause");
            break;
        case 9:
            printf("Ingrese el numero X: ");
            scanf("%d",&elemX);
            BuscarX(Lista,elemX,'D');
            Lista=EliminarDespuesX(Lista,elemX);
            printf("\n\n");
            system("pause");
            break;
        case 10:
            Lista=OrdenarLista(Lista);
            printf("\n\n");
            system("pause");
            break;
        case 11:
            Recorrer(Lista);
            printf("\n\n");
            system("pause");
            break;
        case 12:
            printf("Ingrese el numero X: ");
            scanf("%d",&elemX);
            printf("Dame el nuevo elemento que lo va a sustituir: ");
            scanf("%d",&elem);
            Lista = ModificarElemX(Lista,elemX,elem);
            printf("\n\n");
            system("pause");
            break;
        case 13:
           printf("Elimina elementos duplicados");
           Lista=EliminarDuplicados(Lista);
           Recorrer(Lista);
           printf("\n\n");
           system("pause");
            break;
        case 14:
            break;
        }
    }
```

```
    printf("\nHasta luego\n");
    system("pause");
    return 0;
}
```

diagrama:




case 11: recorrer(Lista)
printf("\n\n")
system("pause")
case 14: break
printf("\nHasta luego\n")
system("pause")
return 0
Lin 166 - Col 5