



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

### Laboratorios de computación salas A y B

*Profesor:* **Manuel Enrique Castañeda Castañeda**

*Asignatura:*

**Estructura de Datos y Algoritmos I**

*Grupo:*

**12**

*No de Práctica(s):*

*Integrante(s):*

- **Lemus Ambrosio Aline Andrea**
- **Sánchez Alvirde Andrés Iván**
- **Rios Morales Manuel**

*No. de Equipo de cómputo  
empleado:*

**-**

*No. de Lista o Brigada:*

**12**

*Semestre:*

**2021-2**

*Fecha de entrega:*

**15 de agosto de 2021**

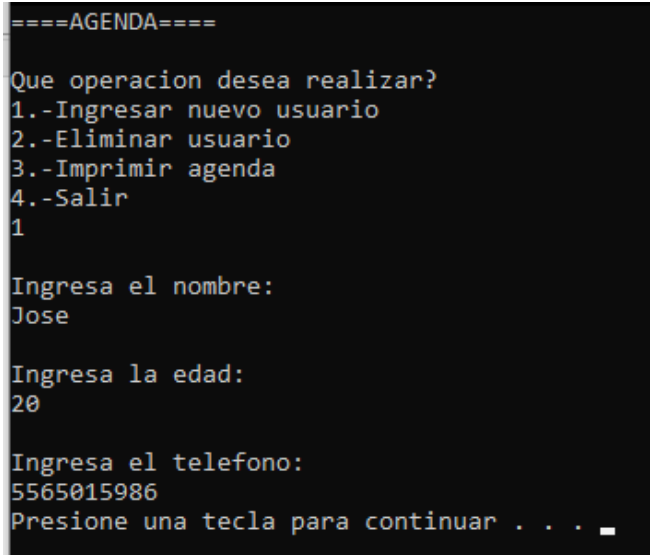
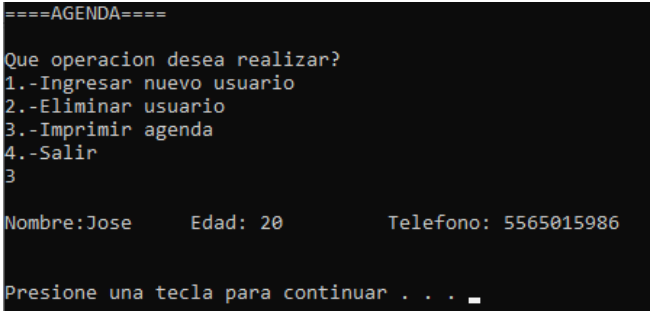
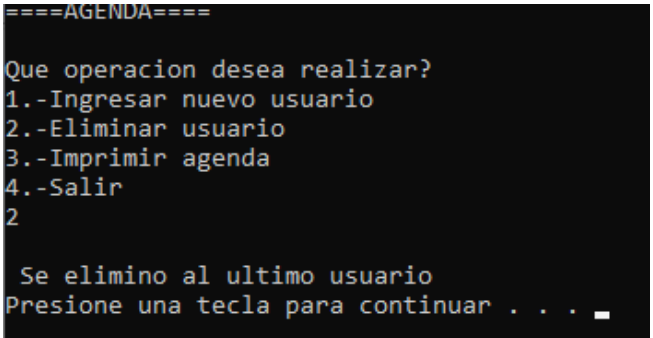
*Observaciones:*

**Calificación:**



## Ejercicios

1. Implementar los siguientes programas usando Pila o Cola simple
  - a. Implementar una agenda de cuates con nombre, edad y teléfono. Almacenar los datos en una pila (agregar, eliminar y mostrar).

CÓDIGO FUENTE	EJECUCIÓN
<pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; struct nodo{     int edad;     long long telefono;     char nombre[50];     nodo *siguiente; }; nodo *pila; void push(); void pop(); void imprimir(nodo *); main(){     int op;     do{         printf("====AGENDA====\n");         printf("\nQue operacion desea realizar?\n");         ;         printf("1.-Ingresar nuevo usuario\n");         printf("2.-Eliminar usuario\n");         printf("3.-Imprimir agenda\n");         printf("4.-Salir\n");         scanf("%d",&amp;op);         switch(op)         {             case 1: push();system("pause");             case 2: pop();system("pause");             case 3:             imprimir(pila);system("pause"); system("cls"); break;             case 4: printf("\nAdios\n");break;         }     }while(op&lt;4); }  void push(){     nodo *nuevo=(nodo*)malloc(sizeof(nodo));     nuevo-&gt;siguiente=pila;     printf("\nIngresa el nombre:\n");     for(int i=0;i&lt;1;i++){         scanf("%s",&amp;nuevo-&gt;nombre[i]);     }     printf("\nIngresa la edad:\n");     scanf("%d",&amp;nuevo-&gt;edad);</pre>	 <pre>====AGENDA==== Que operacion desea realizar? 1.-Ingresar nuevo usuario 2.-Eliminar usuario 3.-Imprimir agenda 4.-Salir 1 Ingresa el nombre: Jose Ingresa la edad: 20 Ingresa el telefono: 5565015986 Presione una tecla para continuar . . .</pre>  <pre>====AGENDA==== Que operacion desea realizar? 1.-Ingresar nuevo usuario 2.-Eliminar usuario 3.-Imprimir agenda 4.-Salir 3 Nombre:Jose      Edad: 20      Telefono: 5565015986 Presione una tecla para continuar . . .</pre>  <pre>====AGENDA==== Que operacion desea realizar? 1.-Ingresar nuevo usuario 2.-Eliminar usuario 3.-Imprimir agenda 4.-Salir 2 Se elimino al ultimo usuario Presione una tecla para continuar . . .</pre>

```

        printf("\nIngresa el telefono:\n");
        scanf("%lld",&nuevo->telefono);
        pila=nuevo;
    }

void pop(){
    nodo *elim;
    int x;
    char y;
    if(pila==NULL){
        printf("\nLa pila se encuentra vacia,
imposible eliminar\n");
    }
    else{
        x=pila->edad+pila->telefono;
        y=pila->nombre[50];
        elim=pila;
        pila=pila->siguiente;
        free(elim);
        printf("\n Se elimino al ultimo usuario\n\a");
    }
}

void imprimir(nodo *){
    nodo *indice=pila;
    while(indice!=NULL){
        printf("\nNombre:%s\tEdad: %d\t Telefono:
%lld\n",indice->nombre,indice->edad,indice->telefono);
        indice= indice->siguiente;
        printf("\n");
    }
    printf("\n");
}

```

====AGENDA====

Que operacion desea realizar?

1.-Ingresar nuevo usuario

2.-Eliminar usuario

3.-Imprimir agenda

4.-Salir

3

Presione una tecla para continuar . . .

- b. Elaborar un programa que evalúe un número, si es par agregarlo a una pila, si es impar, agregarlo a una cola

CÓDIGO FUENTE	EJECUCIÓN
<pre> #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt;  struct nodo{ int dato; nodo *siguiente; }; nodo *fin=NULL; nodo *pila; nodo *inicio=NULL; void pushp(); void mostrcol(); void mostrpil(); main(){     int op, op1;     do{         printf("%cQu%c operacion deseas realizar?\n", 168,130 );         printf("1.- Insertar dato\n");         printf("2.-Mostrar cola\n");         printf("3.-Mostrar pila\n");         printf("4.-Mostrar ambas\n");         printf("5.-Salir\n");         scanf("%d",&amp;op);         switch(op){             case 1:                 pushp();                 break;             case 2:mostrcol();                 system ("pause");                 system ("cls");                 break;             case 3:mostrpil();                 system ("pause");                 system ("cls");                 break;             case 4:                 printf("\nLos datos ingresados de la pila son:\n");                 mostrpil();                 printf("\n");                 printf("\nLos datos ingresados de la cola son:\n");                 mostrcol();                 system ("pause");                 system ("cls");                 break;             case 5:                 printf("\n Hasta luego"); </pre>	<pre> ¿Qué operacion deseas realizar? 1.- Insertar dato 2.-Mostrar cola 3.-Mostrar pila 4.-Mostrar ambas 5.-Salir 1 Cuantos numeros quiere agregar? 6  Ingrese el dato: 1  Ingrese el dato: 2  Ingrese el dato: 3  Ingrese el dato: 4  Ingrese el dato: 5  Ingrese el dato: 6 Presione una tecla para continuar . . . _  ¿Qué operacion deseas realizar? 1.- Insertar dato 2.-Mostrar cola 3.-Mostrar pila 4.-Mostrar ambas 5.-Salir 4  Los datos ingresados de la pila son: [6]      [4]      [2]  Los datos ingresados de la cola son: [1]      [3]      [5] Presione una tecla para continuar . . . </pre>

```

        break;
        default:
        printf("opci%cn, no es valida\n",
162);
        break;
    }
}while(op<5);
}

void mostrpil(){
    nodo *indice = pila;
    while(indice!=NULL){
        printf("[%d]\t",indice->dato);
        indice = indice->siguiente;
    }
    printf("\n");
}

void mostrcol(){
    nodo *indice=inicio;
    while(indice!=NULL){
        printf("[%d]\t",indice->dato);
        indice=indice->siguiente;
    }
    printf("\n");
}

void pushp(){
    int x,n;
    printf("Cuantos numeros quiere agregar?\n");
    scanf("%d",&n);
    for (int i=1;i<=n;i++){
        printf("\nIngresa el dato:\n");
        scanf("%d",&x);
        if(x%2==0 || x==0){
            nodo *nuevo=(nodo*)malloc(sizeof(nodo));
            nuevo->siguiente=pila;
            nuevo->dato=x;
            pila = nuevo;
        }
        else{
            nodo *nuevo =
(nodo*)malloc(sizeof(nodo));
            nuevo->dato = x;
            nuevo->siguiente = NULL;
            if(inicio==NULL){
                inicio=nuevo;
                fin=nuevo;
            }
            else{
                fin->siguiente=nuevo;
                fin=nuevo;
            }
        }
    }
}

system ("pause");

```

<pre> system ("cls"); } </pre>	
--------------------------------	--

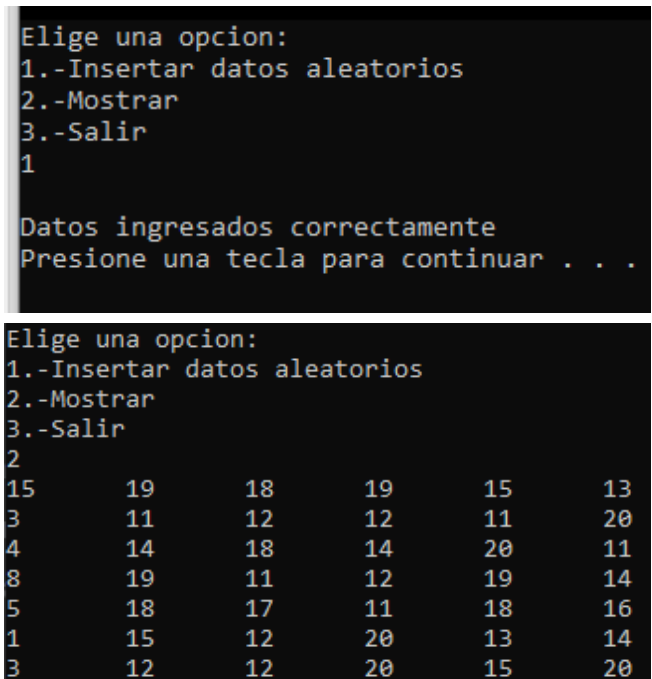
- c. Elaborar un programa que al eliminar un elemento de una cola este se agregue en automático a una pila

CÓDIGO FUENTE	EJECUCIÓN

- d. Elaborar un programa que simule la cola impresión, habrá dos colas una trabajo normal y otra trabajo urgente, el usuario indicará esta situación, si es normal se almacena en una cola si es urgente otra cola

CÓDIGO FUENTE	EJECUCIÓN

2. Implementar los siguientes programas usando Cola circular o cola doble
- a. Implementar una cola doble que se llenará con 100 números aleatorios entre el 10 y 20

CÓDIGO FUENTE	EJECUCIÓN																																										
<pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;time.h&gt; struct nodo{     int dato;     nodo *siguiente;     nodo *anterior; };  nodo *primero=NULL; nodo *ultimo=NULL; void insertarNodo(); void mostrarLista(); int main(){     int op;     do{         printf("Elige una opcion:\n");         printf("1.-Insertar datos aleatorios\n");         printf("2.-Mostrar\n");         printf("3.-Salir\n");         scanf("%d",&amp;op);         switch(op){             case 1:</pre>	 <p>Elige una opcion: 1.-Insertar datos aleatorios 2.-Mostrar 3.-Salir 1</p> <p>Datos ingresados correctamente Presione una tecla para continuar . . .</p> <p>Elige una opcion: 1.-Insertar datos aleatorios 2.-Mostrar 3.-Salir 2</p> <table><tr><td>15</td><td>19</td><td>18</td><td>19</td><td>15</td><td>13</td></tr><tr><td>3</td><td>11</td><td>12</td><td>12</td><td>11</td><td>20</td></tr><tr><td>4</td><td>14</td><td>18</td><td>14</td><td>20</td><td>11</td></tr><tr><td>8</td><td>19</td><td>11</td><td>12</td><td>19</td><td>14</td></tr><tr><td>5</td><td>18</td><td>17</td><td>11</td><td>18</td><td>16</td></tr><tr><td>1</td><td>15</td><td>12</td><td>20</td><td>13</td><td>14</td></tr><tr><td>3</td><td>12</td><td>12</td><td>20</td><td>15</td><td>20</td></tr></table>	15	19	18	19	15	13	3	11	12	12	11	20	4	14	18	14	20	11	8	19	11	12	19	14	5	18	17	11	18	16	1	15	12	20	13	14	3	12	12	20	15	20
15	19	18	19	15	13																																						
3	11	12	12	11	20																																						
4	14	18	14	20	11																																						
8	19	11	12	19	14																																						
5	18	17	11	18	16																																						
1	15	12	20	13	14																																						
3	12	12	20	15	20																																						

```

        insertarNodo();
        system("pause");
        system("cls");
    break;
    case 2:
        mostrarLista();
        system("pause");
        system("cls");
    break;
    case 3:
        printf("\nVuelva pronto\n");
    break;
}
}while(op<3);
}

void insertarNodo(){
    srand(time(NULL));
    for (int x=1;x<=100;x++){
        nodo *nuevo=(nodo*)malloc(sizeof(nodo));
        nuevo->dato=rand()%10+11;
        if(primeros==NULL){
            primero=nuevo;
            primero->siguiente=primero;
            ultimo=primero;
            primero->anterior=ultimo;
        }
        else{
            ultimo->siguiente=nuevo;
            nuevo->siguiente=primero;
            nuevo->anterior=ultimo;
            ultimo=nuevo;
            primero->anterior=ultimo;
        }
    }
    printf("\nDatos ingresados correctamente\n\n");
}

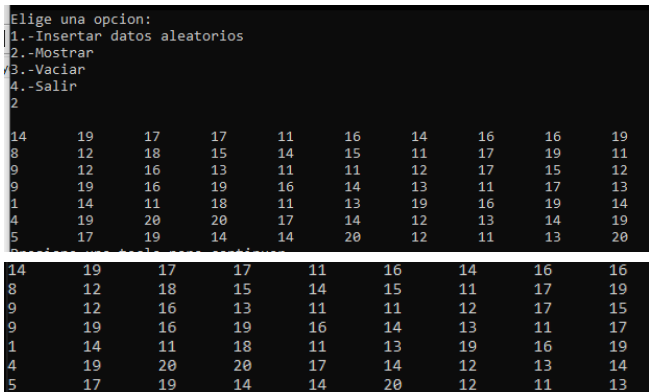
void mostrarLista(){
    nodo *actual=(nodo*)malloc(sizeof(nodo));
    actual=primero;
    if(actual!=NULL){
        do{
            printf("%d\t",actual->dato);
            actual=actual->siguiente;
        }while(actual!=primero);
        printf("\n");
    }
    else{
        printf("La lista esta vacia\n");
    }
}

```

15	17	15	14	15	15	12	12	17
14	11	19	18	19	11	19	13	20
14	17	14	16	11	14	13	15	17
15	16	13	15	20	15	14	19	16
19	15	18	14	15	18	15	11	18
12	14	19	19	17	17	19	17	19
16	15	17	19					



- b. A partir del ejercicio 1 se vaciará, eliminando el número mayor comparado entre el número que esta al inicio y el número que esta al final

CÓDIGO FUENTE	EJECUCIÓN																																																																																																																																												
<pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;time.h&gt; struct nodo{     int dato;     nodo *siguiente;     nodo *anterior; };  nodo *primero=NULL; nodo *ultimo=NULL; void insertarNodo(); void mostrarLista(); void eliminarNodo(); int main(){     int op;     do{         printf("Elige una opcion:\n");         printf("1.-Insertar datos aleatorios\n");         printf("2.-Mostrar\n");         printf("3.-Vaciar\n");         printf("4.-Salir\n");         scanf("%d",&amp;op);         switch(op){             case 1:                 insertarNodo();                 system("pause");                 system("cls");              break;             case 2:                 mostrarLista();                 system("pause");                 system("cls");              break;             case 3:                 eliminarNodo();              break;             case 4:                 printf("\nVuelva pronto\n");              break;         }     }while(op&lt;4); }  void insertarNodo(){     srand(time(NULL));     for (int x=1;x&lt;=100;x++){         nodo *nuevo=(nodo*)malloc(sizeof(nodo));</pre>	 <p>The screenshot shows the program's execution. It starts with a menu where option 2 ('Mostrar') is selected. This results in a 10x10 grid of random numbers. The numbers in the grid are as follows:</p> <table><tr><td>14</td><td>19</td><td>17</td><td>17</td><td>11</td><td>16</td><td>14</td><td>16</td><td>16</td><td>19</td></tr><tr><td>8</td><td>12</td><td>18</td><td>15</td><td>14</td><td>15</td><td>11</td><td>17</td><td>19</td><td>11</td></tr><tr><td>9</td><td>12</td><td>16</td><td>13</td><td>11</td><td>11</td><td>12</td><td>17</td><td>15</td><td>12</td></tr><tr><td>9</td><td>19</td><td>16</td><td>19</td><td>16</td><td>14</td><td>13</td><td>11</td><td>17</td><td>13</td></tr><tr><td>1</td><td>14</td><td>11</td><td>18</td><td>11</td><td>13</td><td>19</td><td>16</td><td>19</td><td>14</td></tr><tr><td>4</td><td>19</td><td>20</td><td>20</td><td>17</td><td>14</td><td>12</td><td>13</td><td>14</td><td>19</td></tr><tr><td>5</td><td>17</td><td>19</td><td>14</td><td>14</td><td>20</td><td>12</td><td>11</td><td>13</td><td>20</td></tr><tr><td>14</td><td>19</td><td>17</td><td>17</td><td>11</td><td>16</td><td>14</td><td>16</td><td>16</td><td>16</td></tr><tr><td>8</td><td>12</td><td>18</td><td>15</td><td>14</td><td>15</td><td>11</td><td>17</td><td>19</td><td>19</td></tr><tr><td>9</td><td>12</td><td>16</td><td>13</td><td>11</td><td>11</td><td>12</td><td>17</td><td>15</td><td>15</td></tr><tr><td>9</td><td>19</td><td>16</td><td>19</td><td>16</td><td>14</td><td>13</td><td>11</td><td>17</td><td>17</td></tr><tr><td>1</td><td>14</td><td>11</td><td>18</td><td>11</td><td>13</td><td>19</td><td>16</td><td>19</td><td>19</td></tr><tr><td>4</td><td>19</td><td>20</td><td>20</td><td>17</td><td>14</td><td>12</td><td>13</td><td>14</td><td>14</td></tr><tr><td>5</td><td>17</td><td>19</td><td>14</td><td>14</td><td>20</td><td>12</td><td>11</td><td>13</td><td>13</td></tr></table>	14	19	17	17	11	16	14	16	16	19	8	12	18	15	14	15	11	17	19	11	9	12	16	13	11	11	12	17	15	12	9	19	16	19	16	14	13	11	17	13	1	14	11	18	11	13	19	16	19	14	4	19	20	20	17	14	12	13	14	19	5	17	19	14	14	20	12	11	13	20	14	19	17	17	11	16	14	16	16	16	8	12	18	15	14	15	11	17	19	19	9	12	16	13	11	11	12	17	15	15	9	19	16	19	16	14	13	11	17	17	1	14	11	18	11	13	19	16	19	19	4	19	20	20	17	14	12	13	14	14	5	17	19	14	14	20	12	11	13	13
14	19	17	17	11	16	14	16	16	19																																																																																																																																				
8	12	18	15	14	15	11	17	19	11																																																																																																																																				
9	12	16	13	11	11	12	17	15	12																																																																																																																																				
9	19	16	19	16	14	13	11	17	13																																																																																																																																				
1	14	11	18	11	13	19	16	19	14																																																																																																																																				
4	19	20	20	17	14	12	13	14	19																																																																																																																																				
5	17	19	14	14	20	12	11	13	20																																																																																																																																				
14	19	17	17	11	16	14	16	16	16																																																																																																																																				
8	12	18	15	14	15	11	17	19	19																																																																																																																																				
9	12	16	13	11	11	12	17	15	15																																																																																																																																				
9	19	16	19	16	14	13	11	17	17																																																																																																																																				
1	14	11	18	11	13	19	16	19	19																																																																																																																																				
4	19	20	20	17	14	12	13	14	14																																																																																																																																				
5	17	19	14	14	20	12	11	13	13																																																																																																																																				

```

nuevo->dato=rand()%10+11;
if(primeros==NULL){
    primero=nuevo;
    primero->siguiente=primero;
    ultimo=primero;
    primero->anterior=ultimo;
}
else{
    ultimo->siguiente=nuevo;
    nuevo->siguiente=primero;
    nuevo->anterior=ultimo;
    ultimo=nuevo;
    primero->anterior=ultimo;
}
}
printf("\nDatos ingresados correctamente\n\n");
}

void mostrarLista(){
    nodo *actual=(nodo*)malloc(sizeof(nodo));
    actual=primero;
    printf("\n");
    if(actual!=NULL){
        do{
            printf("%d\t",actual->dato);
            actual=actual->siguiente;
        }while(actual!=primero);
        printf("\n");
    }
    else{
        printf("La lista esta vacia\n");
    }
}

void eliminarNodo(){
    int nodoBuscado,encontrado=0;
    nodo *actual=(nodo*)malloc(sizeof(nodo));
    actual=primero;
    nodo *ant=(nodo*)malloc(sizeof(nodo));
    ant=NULL;
    nodo *final=(nodo*)malloc(sizeof(nodo));
    final=ultimo;
    if(primero!=NULL&&final!=NULL){
        if(actual->dato > final->dato){
            if(actual==primero){
                primero=primero->siguiente;
                primero->anterior=ultimo;
                ultimo->siguiente=primero;
            }
            else if(actual==ultimo){
                ultimo=ant;
            }
        }
    }
}

```

```

ultimo->siguiente=primero;

primero->anterior=ultimo;
    }
    else{

ant->siguiente=actual->siguiente;

actual->siguiente->anterior=ant;
    }
    printf("Nodo eliminado\n");
    encontrado=1;
    ant=actual;
    actual=actual->siguiente;
}
else{
    if(final==ultimo){

ultimo=ultimo->anterior;

ultimo->siguiente=primero;

primero->anterior=ultimo;
    }
    else if(actual==ultimo){
        actual=ant;

actual->anterior=ultimo;

final->siguiente=primero;
    }
    else{

ant->anterior=final->anterior;

final->anterior->siguiente=ant;
    }
    printf("Nodo eliminado\n");
    encontrado=1;
    ant=final;
    final=final->anterior;
}
if(encontrado==0){
    printf("Son iguales\n");
    printf("%d\n",primero->dato);
    printf("%d\n",ultimo->dato);
}
else{
    free(ant);
}
}
else{
    printf("La lista se encuentra vacia\n");

```

<pre>         }     } </pre>	
------------------------------	--

- c. Implementar una cola circular para mostrar un letrero de nombres, después último se mostrará el primero

CÓDIGO FUENTE	EJECUCIÓN
<pre> #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt;  struct nodo{     char nombre[50];     nodo *siguiente;     nodo *anterior; };  nodo *primero=NULL; nodo *ultimo=NULL; void insertarNodo(); void mostrarLista(); void mostrarListaAR(); int main(){     int op;     do{         printf("Elige una opcion:\n");         printf("1.-Insertar nombre\n");         printf("2.-Mostrar nombre(s)\n");         printf("3.-Mostrar ultimo como primero\n");         printf("4.-Salir\n");         scanf("%d",&amp;op);         switch(op){             case 1:                 insertarNodo();                 system("pause");                 system("cls");                 break;             case 2:                 mostrarLista();                 system("pause");                 system("cls");                 break;             case 3:                 mostrarListaAR();                 system("pause");                 system("cls");                 break;             case 4:                 printf("\nVuelva pronto\n");                 break;         }     } } </pre>	<pre> Elige una opcion: 1.-Insertar nombre 2.-Mostrar nombre(s) 3.-Mostrar ultimo como primero 4.-Salir 1 Ingrese el nombre: Pedro  Dato ingresado correctamente Presione una tecla para continuar . . . </pre> <pre> Elige una opcion: 1.-Insertar nombre 2.-Mostrar nombre(s) 3.-Mostrar ultimo como primero 4.-Salir 2 [Luis] [Jose] [Manuel] [Pedro] Presione una tecla para continuar . . . </pre> <pre> Elige una opcion: 1.-Insertar nombre 2.-Mostrar nombre(s) 3.-Mostrar ultimo como primero 4.-Salir 3 [Pedro] [Manuel] [Jose] [Luis] Presione una tecla para continuar . . . </pre>

```

        }while(op<4);
    }

void insertarNodo(){
    nodo *nuevo=(nodo*)malloc(sizeof(nodo));
    printf("Ingrese el nombre:\n");
    for(int i=0;i<=0;i++){
        scanf("%s",&nuevo->nombre);
    }
    if(primeros==NULL){
        primeros=nuevo;
        primeros->siguiente=primeros;
        ultimo=primeros;
        primeros->anterior=ultimo;
    }
    else{
        ultimo->siguiente=nuevo;
        nuevo->siguiente=primeros;
        nuevo->anterior=ultimo;
        ultimo=nuevo;
        primeros->anterior=ultimo;
    }
    printf("\nDato ingresado correctamente\n");
}

void mostrarLista(){
    nodo *actual=(nodo*)malloc(sizeof(nodo));
    actual=primeros;
    if(actual!=NULL){
        do{
            printf("[%s]\t",actual->nombre);
            actual=actual->siguiente;
        }while(actual!=primeros);
        printf("\n");
    }
    else{
        printf("La lista esta vacia\n");
    }
}

void mostrarListaAR(){
    nodo *actual=(nodo*)malloc(sizeof(nodo));
    actual=ultimo;
    if(actual!=NULL){
        do{
            printf("[%s]\t",actual->nombre);
            actual=actual->anterior;
        }while(actual!=ultimo);
        printf("\n");
    }
    else{
        printf("La lista esta vacia\n");
    }
}

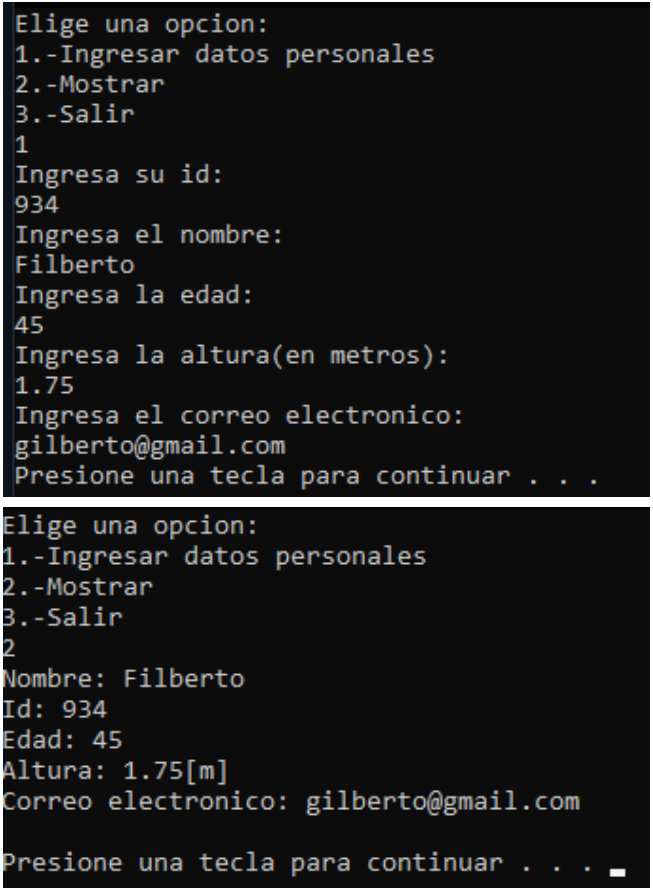
```

--	--

- d. Implementar una cola circular para avanzar en un tablero circular de 30 casillas, los dados se tiran aleatoriamente

CÓDIGO FUENTE	EJECUCIÓN

3. Implementar los siguientes programas usando Lista
- a. Capturar los datos de una persona: id, nombre, edad, altura y correo electrónico.

CÓDIGO FUENTE	EJECUCIÓN
<pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt;  struct nodo{     int id,edad;     float altura;     char nombre[50],correo[50];     nodo *siguiente; }; nodo *lista; void insertar(); void mostrar(nodo *siguiente); int main(){     int op;     do{         printf("Elige una opcion:\n");         printf("1.-Ingresar datos personales\n");         printf("2.-Mostrar\n");         printf("3.-Salir\n");         scanf("%d",&amp;op);         switch(op){             case 1:                 insertar();                 system("pause");                 system("cls");                 break;             case 2:                 mostrar(lista);                 system("pause");                 system("cls");                 break;             case 3:                 printf("\nVuelva pronto\n");                 break;         }     }while(op&lt;3);</pre>	 <pre>Elige una opcion: 1.-Ingresar datos personales 2.-Mostrar 3.-Salir 1 Ingresa su id: 934 Ingresa el nombre: Filberto Ingresa la edad: 45 Ingresa la altura(en metros): 1.75 Ingresa el correo electronico: gilberto@gmail.com Presione una tecla para continuar . . .  Elige una opcion: 1.-Ingresar datos personales 2.-Mostrar 3.-Salir 2 Nombre: Filberto Id: 934 Edad: 45 Altura: 1.75[m] Correo electronico: gilberto@gmail.com Presione una tecla para continuar . . .</pre>

```

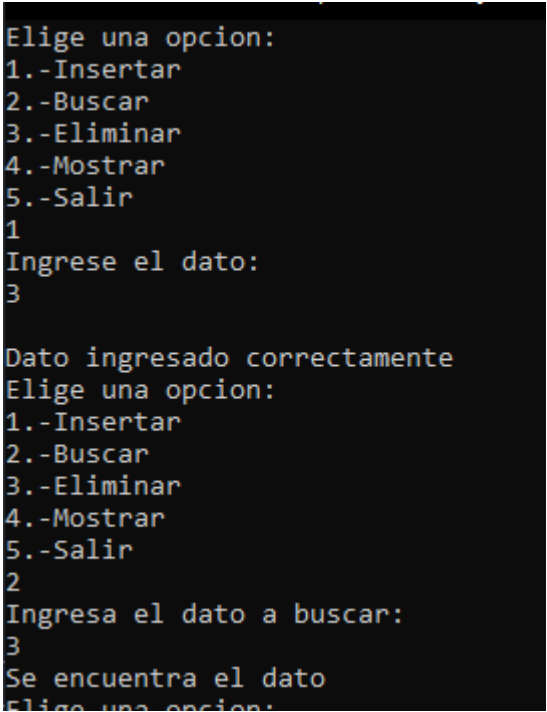
}

void insertar(){
    nodo *q=(nodo*)malloc(sizeof(nodo));
    q->siguiente=lista;
    printf("Ingresa su id:\n");
    scanf("%d",&q->id);
    printf("Ingresa el nombre:\n");
    for(int i=0;i<1;i++){
        scanf("%s",&q->nombre[i]);
    }
    printf("Ingresa la edad:\n");
    scanf("%d",&q->edad);
    printf("Ingresa la altura(en metros):\n");
    scanf("%f",&q->altura);
    printf("Ingresa el correo electronico:\n");
    for(int i=0;i<1;i++){
        scanf("%s",&q->correo[i]);
    }
    lista=q;
}

void mostrar(nodo *siguiente){
    nodo *indice=lista;
    if(indice==NULL){
        printf("Lista vacia\n");
    }
    else{
        while(indice!=NULL){
            printf("Nombre: %s\nId:
%d\nEdad: %d\nAltura: %.2f[m]\nCorreo electronico:
%s\n",indice->nombre,indice->id,indice->edad,indice->altura,
indice->correo);
            indice=indice->siguiente;
            printf("\n");
        }
    }
}

```

b. Implementar la función buscar en las funciones realizadas en clase.

CÓDIGO FUENTE	EJECUCIÓN
<pre>#include &lt;stdio.h&gt;  #include &lt;stdlib.h&gt;  struct nodo{      int dato;      nodo *siguiente;      nodo *anterior;  };  nodo *primero=NULL;  nodo *ultimo=NULL;  void insertarNodo();  void buscarNodo();  void modificarNodo();  void eliminarNodo();  void mostrarLista();  int main(){      int op;      do{          printf("Elige una opcion:\n");          printf("1.-Insertar\n");          printf("2.-Buscar\n");          printf("3.-Eliminar\n");          printf("4.-Mostrar\n");          printf("5.-Salir\n");          scanf("%d",&amp;op);</pre>	



```

switch(op){
    case 1:
        insertarNodo();
        break;
    case 2:
        buscarNodo();
        break;
    case 3:
        eliminarNodo();
        break;
    case 4:
        mostrarLista();
        break;
    case 5:
        printf("\nVuelva
pronto\n");
        break;
}
}while(op<5);
}

```

```

void insertarNodo(){
    nodo *nuevo=(nodo*)malloc(sizeof(nodo));
    printf("Ingrese el dato:\n");
    scanf("%d",&nuevo->dato);
    if(primeros==NULL){
        primeros=nuevo;
        primeros->siguiente=primeros;
    }
}

```

```
        ultimo=primero;

        primero->anterior=ultimo;

    }

    else{

        ultimo->siguiente=nuevo;

        nuevo->siguiente=primero;

        nuevo->anterior=ultimo;

        ultimo=nuevo;

        primero->anterior=ultimo;

    }

    printf("\nDato ingresado correctamente\n");

}

void buscarNodo(){

    int nodoBuscado,encontrado=0;

    nodo *actual=(nodo*)malloc(sizeof(nodo));

    actual=primero;

    printf("Ingresa el dato a buscar:\n");

    scanf("%d",&nodoBuscado);

    if(primero!=NULL){

        do{

            if(actual->dato==nodoBuscado){

                printf("Se encuentra\n\n");

                encontrado=1;

            }

            actual=actual->siguiente;
```

```
        }while(encontrado==0 && actual!=primero);

        if(encontrado==0){

                printf("No se encuentra el
dato\n");

        }

    }

    else{

        printf("La lista se encuentra vacia\n");

    }

}
```

```
void eliminarNodo(){

    int nodoBuscado,encontrado=0;

    nodo *actual=(nodo*)malloc(sizeof(nodo));

    actual=primero;

    nodo *ant=(nodo*)malloc(sizeof(nodo));

    ant=NULL;

    printf("Ingresa el dato a buscar:\n");

    scanf("%d",&nodoBuscado);

    if(primero!=NULL){

        do{

            if(actual->dato==nodoBuscado){

                printf("Se encuentra
el dato\n");

                if(actual==primero){

                    primero=primero->siguiente;

                    primero->anterior=ultimo;

                }

            }

        }while(actual!=NULL);

    }

}
```

```

ultimo->siguiente=primero;
    }
    else
if(actual==ultimo){
    ultimo=ant;

ultimo->siguiente=primero;

primero->anterior=ultimo;
    }
    else{

ant->siguiente=actual->siguiente;

actual->siguiente->anterior=ant;
    }

printf("Elemento
eliminado eliminado\n");

    encontrado=1;
}

ant=actual;

actual=actual->siguiente;
}while(encontrado==0 && actual!=primero);

if(encontrado==0){

printf("No se encuentra el
dato\n");

}

else{

free(ant);

}

```

<pre>         }          else{              printf("La lista se encuentra vacia\n");          }      }  void mostrarLista(){      nodo *actual=(nodo*)malloc(sizeof(nodo));      actual=primero;      if(actual!=NULL){          do{              printf("[%d]\t",actual-&gt;dato);              actual=actual-&gt;siguiente;          }while(actual!=primero);          printf("\n");      }      else{          printf("La lista esta vacia\n");      }  } </pre>	
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

**c. Implementar una lista ligada doble**

CÓDIGO FUENTE	EJECUCIÓN
---------------	-----------

```

#include <stdio.h>
#include <stdlib.h>

struct nodo{
    int dato;
    nodo *siguiente;
    nodo *anterior;
};

nodo *primero=NULL;
nodo *ultimo=NULL;
void insertarNodo();
void buscarNodo();
void eliminarNodo();
void mostrarLista();
int main(){
    int op;
    do{
        printf("Elige una opcion:\n");
        printf("1.-Insertar\n");
        printf("2.-Buscar\n");
        printf("3.-Eliminar\n");
        printf("4.-Mostrar\n");
        printf("5.-Salir\n");
        scanf("%d",&op);
        switch(op){
            case 1:
                insertarNodo();
                break;
            case 2:
                buscarNodo();
                break;
            case 3:
                eliminarNodo();
                break;
            case 4:
                mostrarLista();
                break;
            case 5:
                printf("\nVuelva
pronto\n");
                break;
        }
    }while(op<5);
}

void insertarNodo(){
    nodo *nuevo=(nodo*)malloc(sizeof(nodo));
    printf("Ingrese el dato:\n");

```

```

Elige una opcion:
1.-Insertar
2.-Buscar
3.-Eliminar
4.-Mostrar
5.-Salir
1
Ingrese el dato:
4

Dato ingresado correctamente
Elige una opcion:
1.-Insertar
2.-Buscar
3.-Eliminar
4.-Mostrar
5.-Salir
4
[4]

```

```

scanf("%d",&nuevo->dato);
if(primerο==NULL){
    primero=nuevo;
    primero->siguiente=primero; //ligarse
    ultimo=primero;
    primero->anterior=ultimo;
}
else{
    ultimo->siguiente=nuevo;
    nuevo->siguiente=primero;
    nuevo->anterior=ultimo;
    ultimo=nuevo;
    primero->anterior=ultimo;
}
printf("\nDato ingresado correctamente\n");
}

void buscarNodo(){
    int nodoBuscado,encontrado=0;
    nodo *actual=(nodo*)malloc(sizeof(nodo));
    actual=primero;
    printf("Ingresa el dato a buscar:\n");
    scanf("%d",&nodoBuscado);
    if(primero!=NULL){
        do{

if(actual->dato==nodoBuscado){
                printf("Se encuentra el
dato\n");
                encontrado=1;
            }
            actual=actual->siguiente;
        }while(encontrado==0 &&
actual!=primero);
        if(encontrado==0){
            printf("No se encuentra el
dato\n");
        }
    }
    else{
        printf("La lista se encuentra vacia\n");
    }
}

void eliminarNodo(){
    int nodoBuscado,encontrado=0;
    nodo *actual=(nodo*)malloc(sizeof(nodo));
    actual=primero;
    nodo *ant=(nodo*)malloc(sizeof(nodo));

```

```

ant=NULL;
printf("Ingresa el dato a buscar:\n");
scanf("%d",&nodoBuscado);
if(primer!=NULL){
    do{

if(actual->dato==nodoBuscado){
                                printf("Se encuentra el
dato\n");
                                if(actual==primer){

primer=primer->siguiente;

primer->anterior=ultimo;

ultimo->siguiente=primer;
                                }
                                else if(actual==ultimo){
                                    ultimo=ant;

ultimo->siguiente=primer;

primer->anterior=ultimo;
                                }
                                else{

ant->siguiente=actual->siguiente;

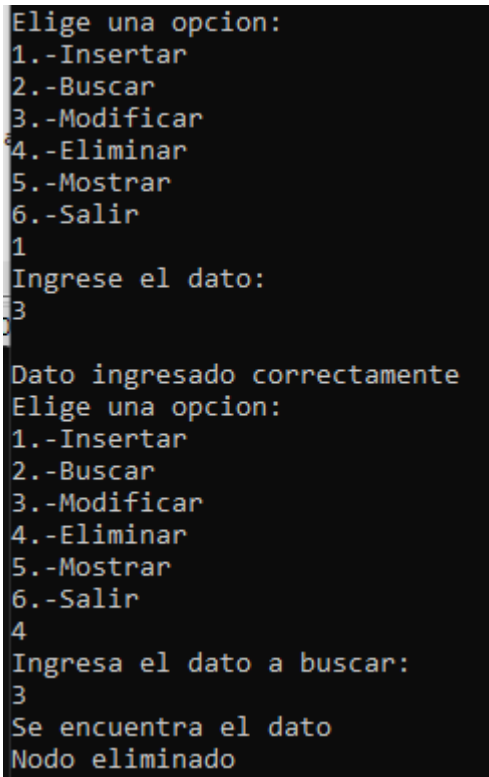
actual->siguiente->anterior=ant;
                                }
                                printf("Nodo
eliminado\n");
                                encontrado=1;
                                }
                                ant=actual;
                                actual=actual->siguiente;
                                }while(encontrado==0 &&
actual!=primer);
                                if(encontrado==0){
                                    printf("No se encuentra el
dato\n");
                                }
                                else{
                                    free(ant);
                                }
                                }
                                else{
                                    printf("La lista se encuentra vacia\n");
                                }
}

```



<pre> }  void mostrarLista(){     nodo *actual=(nodo*)malloc(sizeof(nodo));     actual=primero;     if(actual!=NULL){         do{             printf("[%d]\t",actual-&gt;dato);             actual=actual-&gt;siguiente;         }while(actual!=primero);         printf("\n");     }     else{         printf("La lista esta vacia\n");     } } </pre>	
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

**d. Implementar una lista ligada doble circular**

CÓDIGO FUENTE	EJECUCIÓN
<pre> #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt;  struct nodo{     int dato;     nodo *siguiente;     nodo *anterior; };  nodo *primero=NULL; nodo *ultimo=NULL; void insertarNodo(); void buscarNodo(); void modificarNodo(); void eliminarNodo(); void mostrarLista(); int main(){     int op;     do{         printf("Elige una opcion:\n");         printf("1.-Insertar\n");         printf("2.-Buscar\n");         printf("3.-Modificar\n");         printf("4.-Eliminar\n");         printf("5.-Mostrar\n");         printf("6.-Salir\n");         scanf("%d",&amp;op);         switch(op){             case 1: </pre>	 <pre> Elige una opcion: 1.-Insertar 2.-Buscar 3.-Modificar 4.-Eliminar 5.-Mostrar 6.-Salir 1 Ingrese el dato: 3 Dato ingresado correctamente Elige una opcion: 1.-Insertar 2.-Buscar 3.-Modificar 4.-Eliminar 5.-Mostrar 6.-Salir 4 Ingresa el dato a buscar: 3 Se encuentra el dato Nodo eliminado </pre>

```

                insertarNodo();
            break;
        case 2:
            buscarNodo();
            break;
        case 3:
            modificarNodo();
            break;
        case 4:
            eliminarNodo();
            break;
        case 5:
            mostrarLista();
            break;
        case 6:
            printf("\nVuelva pronto\n");
            break;
    }
}while(op<6);
}

```

```

void insertarNodo(){
    nodo *nuevo=(nodo*)malloc(sizeof(nodo));
    printf("Ingrese el dato:\n");
    scanf("%d",&nuevo->dato);
    if(primeros==NULL){
        primeros=nuevo;
        primeros->siguiente=primeros; //ligarse
        ultimo=primeros;
        primeros->anterior=ultimo;
    }
    else{
        ultimo->siguiente=nuevo;
        nuevo->siguiente=primeros;
        nuevo->anterior=ultimo;
        ultimo=nuevo;
        primeros->anterior=ultimo;
    }
    printf("\nDato ingresado correctamente\n");
}

```

```

void buscarNodo(){
    int nodoBuscado,encontrado=0;
    nodo *actual=(nodo*)malloc(sizeof(nodo));
    actual=primeros;
    printf("Ingresa el dato a buscar:\n");
    scanf("%d",&nodoBuscado);
    if(primeros!=NULL){
        do{
            if(actual->dato==nodoBuscado){
                printf("Se encuentra el
dato\n");
                encontrado=1;
            }
        }
    }
}

```

```

Elige una opcion:
1.-Insertar
2.-Buscar
3.-Modificar
4.-Eliminar
5.-Mostrar
6.-Salir
1
Ingrese el dato:
45

Dato ingresado correctamente
Elige una opcion:
1.-Insertar
2.-Buscar
3.-Modificar
4.-Eliminar
5.-Mostrar
6.-Salir
5
45

```

```

Elige una opcion:
1.-Insertar
2.-Buscar
3.-Modificar
4.-Eliminar
5.-Mostrar
6.-Salir
2
Ingresa el dato a buscar:
45
Se encuentra el dato

```

```

Elige una opcion:
1.-Insertar
2.-Buscar
3.-Modificar
4.-Eliminar
5.-Mostrar
6.-Salir
3
Ingresa el dato a buscar:
45
Se encuentra el dato
Dame el nuevo dato:
17

```

```

Elige una opcion:
1.-Insertar
2.-Buscar
3.-Modificar
4.-Eliminar
5.-Mostrar
6.-Salir
5
17

```

```

        actual=actual->siguiente;
    }while(encontrado==0 && actual!=primero);
    if(encontrado==0){
        printf("No se encuentra el dato\n");
    }
}
else{
    printf("La lista se encuentra vacia\n");
}
}

```

```

void modificarNodo(){
    int nodoBuscado,encontrado=0;
    nodo *actual=(nodo*)malloc(sizeof(nodo));
    actual=primero;
    printf("Ingresa el dato a buscar:\n");
    scanf("%d",&nodoBuscado);
    if(primero!=NULL){
        do{
            if(actual->dato==nodoBuscado){
                printf("Se encuentra el
dato\n");
                printf("Dame el nuevo dato:
\n");
                scanf("%i",&actual->dato);
                encontrado=1;
            }
            actual=actual->siguiente;
        }while(encontrado==0 && actual!=primero);
        if(encontrado==0){
            printf("No se encuentra el dato\n");
        }
    }
    else{
        printf("La lista se encuentra vacia\n");
    }
}

```

```

void eliminarNodo(){
    int nodoBuscado,encontrado=0;
    nodo *actual=(nodo*)malloc(sizeof(nodo));
    actual=primero;
    nodo *ant=(nodo*)malloc(sizeof(nodo));
    ant=NULL;
    printf("Ingresa el dato a buscar:\n");
    scanf("%d",&nodoBuscado);
    if(primero!=NULL){
        do{
            if(actual->dato==nodoBuscado){
                printf("Se encuentra el
dato\n");
                if(actual==primero){
                    primero=primero->siguiente;

```

```

primero->anterior=ultimo;

ultimo->siguiente=primero;
    }
    else if(actual==ultimo){
        ultimo=ant;

ultimo->siguiente=primero;

primero->anterior=ultimo;
    }
    else{

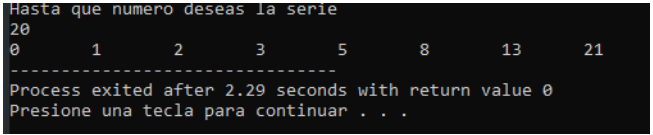
ant->siguiente=actual->siguiente;

actual->siguiente->anterior=ant;
    }
    printf("Nodo eliminado\n");
    encontrado=1;
    }
    ant=actual;
    actual=actual->siguiente;
}while(encontrado==0 && actual!=primero);
if(encontrado==0){
    printf("No se encuentra el dato\n");
}
else{
    free(ant);
}
}
else{
    printf("La lista se encuentra vacia\n");
}
}

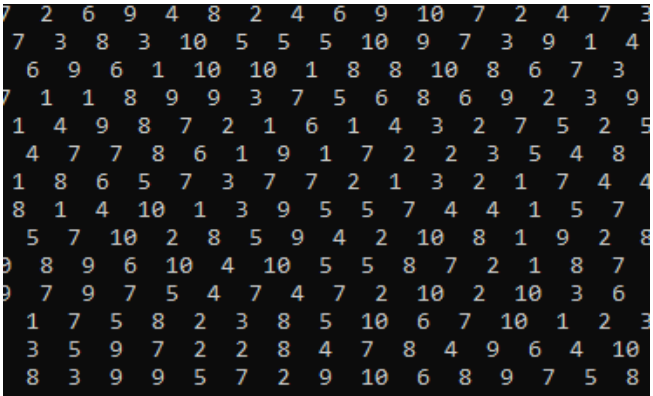
void mostrarLista(){
    nodo *actual=(nodo*)malloc(sizeof(nodo));
    actual=primero;
    if(actual!=NULL){
        do{
            printf("%d\t",actual->dato);
            actual=actual->siguiente;
        }while(actual!=primero);
        printf("\n");
    }
    else{
        printf("La lista esta vacia\n");
    }
}

```

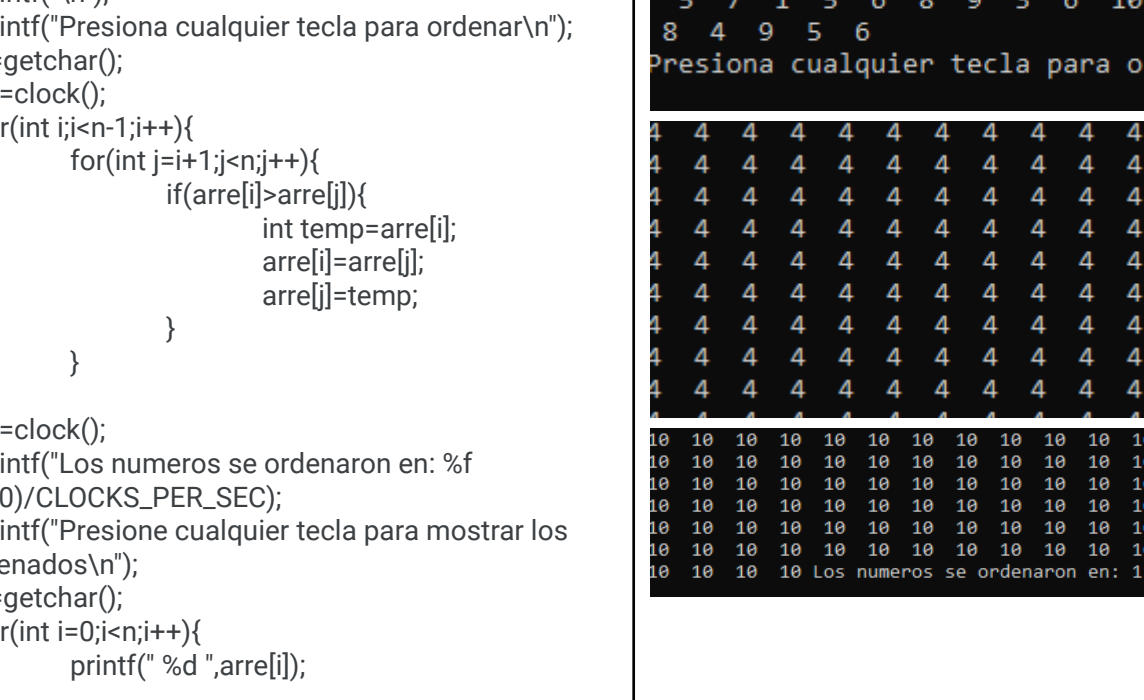
4. Implementar la serie de Fibonacci hasta n dado por el usuario, usando top down

CÓDIGO FUENTE	EJECUCIÓN
<pre>#include&lt;stdio.h&gt; #include&lt;stdlib.h&gt;  int fibonacci(int n); int pedirNumero(); int main(){     fibonacci(pedirNumero());     return 0; }  int fibonacci(int n){     int n1=0,n2=1,n3;     do{         n3=n1+n2;         printf("%d\t",n3);         n1=n2;         n2=n3;     }while(n2&lt;n); }  int pedirNumero(){     int x;     printf("Hasta que numero deseas la serie\n");     scanf("%d",&amp;x);     printf("0\t");     return x; }</pre>	 <p>Hasta que numero deseas la serie 20 0      1      2      3      5      8      13      21 ----- Process exited after 2.29 seconds with return value 0 Presione una tecla para continuar . . .</p>

5. Hacer un arreglo de tamaño grande (más de 100 mil) con el método de la burbuja

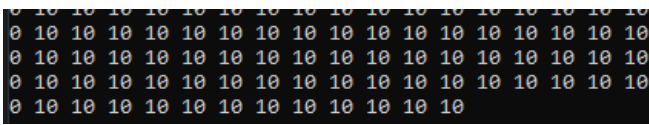
CÓDIGO FUENTE	EJECUCIÓN
<pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;time.h&gt; int main(){     int n=100000;     float t0,t1;     char x;     int arre[n];     srand(time(NULL));     for(int i=0;i&lt;n;i++){         arre[i]=rand()%10+1;         printf(" %d ",arre[i]);     }</pre>	 <p>7 2 6 9 4 8 2 4 6 9 10 7 2 4 7 3 7 3 8 3 10 5 5 5 10 9 7 3 9 1 4 6 9 6 1 10 10 1 8 8 10 8 6 7 3 7 1 1 8 9 9 3 7 5 6 8 6 9 2 3 9 1 4 9 8 7 2 1 6 1 4 3 2 7 5 2 5 4 7 7 8 6 1 9 1 7 2 2 3 5 4 8 1 8 6 5 7 3 7 7 2 1 3 2 1 7 4 4 8 1 4 10 1 3 9 5 5 7 4 4 1 5 7 5 7 10 2 8 5 9 4 2 10 8 1 9 2 8 9 8 9 6 10 4 10 5 5 8 7 2 1 8 7 9 7 9 7 5 4 7 4 7 2 10 2 10 3 6 1 7 5 8 2 3 8 5 10 6 7 10 1 2 3 3 5 9 7 2 2 8 4 7 8 4 9 6 4 10 8 3 9 9 5 7 2 9 10 6 8 9 7 5 8</p>

```
}
printf("\n");
printf("Presiona cualquier tecla para ordenar\n");
x=getchar();
t0=clock();
for(int i;i<n-1;i++){
    for(int j=i+1;j<n;j++){
        if(arre[i]>arre[j]){
            int temp=arre[i];
            arre[i]=arre[j];
            arre[j]=temp;
        }
    }
}
t1=clock();
printf("Los numeros se ordenaron en: %f
[s]\n",(t1-t0)/CLOCKS_PER_SEC);
printf("Presione cualquier tecla para mostrar los
datos ordenados\n");
x=getchar();
for(int i=0;i<n;i++){
    printf(" %d ",arre[i]);
}
}
```



6. Ordenar un arreglo de tamaño grande (más de 100 mil) con el método de Quicksort

CÓDIGO FUENTE	EJECUCIÓN
<pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;time.h&gt; void qs(int arr[],int i,int j); int main(){      int n,i,j;     float t1,t0;     char x;      printf("De que tamaño quieres el arreglo?\n");     scanf("%i",&amp;n);      int *arr;     arr=(int *)malloc(n*sizeof(int));     srand(time(NULL));     for (i=0;i&lt;n;i++){         arr[i]=rand()%10+1;         printf(" %i",arr[i]);     }     qs(arr,0,i);</pre>	<pre>4 1 8 5 1 8 5 4 5 9 8 7 7 8 10 0 8 3 5 7 4 10 7 4 9 2 6 10 6 5 1 9 8 2 3 2 4 6 2 4 3 8 6 8 6 7 7 1 7 7 2 2 6 9 2 9 4 3 4 1 5 4 9 2 7 1 10 7 7 3 9 1 3 6 4 1 5 10 4 3 6 5 4 3 3 9 7 8 2 8 10 1 2 6 8 9 4 9 2 3 7 10 10 8 7 2 5 8 8 10 7 9 6 9 8 7 1 2 7 7 6 5 2 7 2 1 7 7 9 2 10 5 6 5 10 10 3 4 3 10 7 6 6 10 10 8 2 9 8 9 5 7 4 1 4 6 10 9 10 1 10 8 3 9 7 5 1 9 6 8 4 5 9 2 1 3 3 7 8 7 6 6 8 10 10 9 4 9 3 5 8 10 2 10 3 10 9 7 8 1 5 4 1 5 1 6 8 8 6 5 8 3 8 1 3 1 3 10 6 9 6 5 3 6 2 3 7 6 10 5 3 1 9 3 8 2 8 6 3 9 7 6 9 4 7 3 3 10 3 5 10 2 3 8 1 9 9 9 8 10 3 6 6 7 5 10 8 7 7 8 9 1 8 10 5 6 7 1 8 3 8 4 8 2 9 4 10 4 7 10 5 2 3 4 9 4 1 9 5 9 4 1 4 10 4 1 2 2</pre>

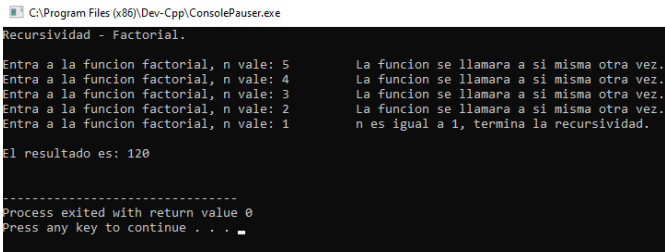
<pre>         printf("\n");         for (i=0;i&lt;n;i++){             printf(" %i",arr[i]);         }     }     void qs(int arr[],int i,int j){          int inicio=i,fin=j,pibote,temp;          pibote=arr[i];//arreglo con pibote, ultimo elemento es el pibote         do{             while(arr[i]&lt;pibote)//elementos menores al pibote                 i=i+1;             while(arr[j]&gt;pibote)                 j=j-1;             if(i&lt;=j){                 temp=arr[i];                 arr[i]=arr[j];                 arr[j]=temp;                 i=i+1;                 j=j-1;             }          }while(i&lt;=j);         if(inicio&lt;j)             qs(arr,inicio,j);         if(i&lt;fin)             qs(arr,i,fin);     } </pre>	
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------

7. Obtener las posibles combinaciones de placas vehiculares en el formato (letraletraletra- numnum), usando fuerza bruta

CÓDIGO FUENTE	EJECUCIÓN
---------------	-----------

<pre>#include&lt;stdio.h&gt; #include&lt;stdlib.h&gt;  struct placa{     char l1,l2,l3;     int p; };  int main(){     placa *mamalon;     mamalon=(placa*)malloc(17576000*sizeof(placa));     int i,j,k,l,m=-1;     for(i='A';i&lt;='Z';i++){         for(j='A';j&lt;='Z';j++){             for(k='A';k&lt;='Z';k++){                 for(l=0;l&lt;=999;l++){                     m++;                     mamalon[m].l1=i;                     mamalon[m].l2=j;                     mamalon[m].l3=k;                     mamalon[m].p=l;                  }             }         }     }      printf("%c%c%c%.3d\n",mamalon[m].l1,mamalon[m].l2,mamalon[m].l3,mamalon[m].p); }</pre>	 
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------

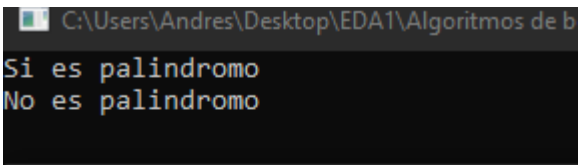
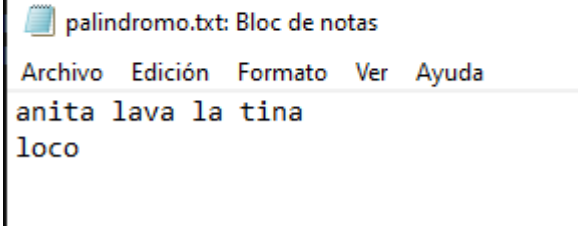
8. Implementar los siguientes programas usando recursividad  
a. Obtener la factorial de un número

CÓDIGO FUENTE	EJECUCIÓN
<pre>#include&lt;stdio.h&gt;  int factorial(int n) {     printf("Entra a la funcion factorial, n vale: %d \t", n);      if (n &gt; 1)     {         //Si n es mayor a 1, entonces se vuelve a llamar la         funcion que multiplica a n * (n -1)         printf("La funcion se llamara a si misma otra vez.\n");         return n * factorial(n-1);     } }</pre>	



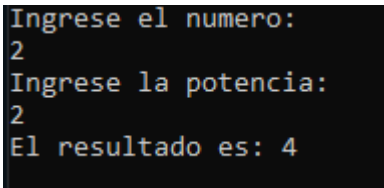
<pre> } else {     //Si n es igual 1, ya no se llama la funcion a si misma para terminar la recursividad.     printf("n es igual a 1, termina la recursividad.\n");     return 1; } }  int main() {     printf("Recursividad - Factorial.\n\n");      int res = factorial(5);     printf("\nEl resultado es: %d \n\n", res);  } </pre>	
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

**b. Determinar si una cadena es palíndromo o no**

CÓDIGO FUENTE	EJECUCIÓN
<pre> #include &lt;stdio.h&gt; #include &lt;string.h&gt; #include &lt;stdlib.h&gt; int espalindromo(char* cadena); int main(){     char cad[30];     FILE *ap=fopen("palindromo.txt","r");//accediendo a un archivo de texto     while(feof(ap)==NULL){          fgets(cad,100,ap);//leyendo cadena del archivo         if(espalindromo(cad)){             printf("Si es palindromo\n");         }else{             printf("No es palindromo\n");         }     } }  int espalindromo(char *cadena){     int j=0,i;     if(strlen(cadena)==1){         return 1;     }      if(cadena[0]==cadena[strlen(cadena)-2]){         for(i=0;i&lt;strlen(cadena)-1;i++){             cadena[i]=cadena[i+1];             j=1; </pre>	 

<pre>         }         for(i=j;i&lt;strlen(cadena)-1;i++){             cadena[i]=NULL;          }         espalindromo(cadena);     }else{         return 0;     } } </pre>	
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

**c. Obtener la n-ésima potencia de un número x**

CÓDIGO FUENTE	EJECUCIÓN
<pre> #include &lt;stdio.h&gt;  int potencia(int b, int e){     if(e&gt;1){         return b*potencia(b,e-1);     }     else{         return b;     } }  int main(){     int x,p;     printf("Ingrese el numero:\n");     scanf("%d",&amp;x);     printf("Ingrese la potencia:\n");     scanf("%d",&amp;p);     int resultado = potencia(x,p);     printf("El resultado es: %d\n",resultado);     return 1; } </pre>	 <pre> Ingrese el numero: 2 Ingrese la potencia: 2 El resultado es: 4 </pre>

**d. Averiguar si un número es perfecto**

CÓDIGO FUENTE	EJECUCIÓN