|  |
| --- |
| **Project Report**  **on**  **Linked List**  **Session 2018-19**    **Submitted By**  **Mohit Chauhan (1803210094)**  **Mourya Pradeep Ramashare (1803210096)**  **Under the guidance of**  **Akhilesh Kumar Srivastava**  **ABES ENGINEERING COLLEGE, GHAZIABAD** |
|  |
| **AFFILIATED TO**  **DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, U.P., LUCKNOW**  **(Formerly UPTU)** |

# STUDENT’S DECLARATION

We hereby declare that the work being presented in this report entitled “**LINKED LIST**” is an authentic record of our own work carried out under the supervision of **Mr. AKHILESH KUMAR SRIVASTAVA.**

The matter embodied in this report has not been submitted by us for the award of any other degree. (font size 12Arial 1.5 Line Spacing)

**Dated:** **Signature of students**

**Mohit Chauhan**

**Mourya Pradeep Ramashare**

**Department: Computer Science and Engineering**

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

**Signature of Supervisor**

**Akhilesh Kumar Srivastava**

**TABLE OF CONTENTS**

Page no

DECLARATION 2

CHAPTER 1 Introduction .4

1 Linked list

1.1 Linear linked list

1.2 Circular linked list

1.3 Doubly linked list

CHAPTER 2 Algorithm 6

2.1 Linear linked list

2.1.1 Traversing a linked list

2.1.2 Print the number of nodes in a linked list

2.1.3 Insert a new node at the beginning

2.1.4 Insert a new node at the end

2.1.5 Insert a new node after a node that has value num

2.1.6 Insert a new node before a node that has value num

2.1.7 Delete the first node

2.1.8 Delete the last node

2.1.9 Delete the node after a given node

2.2 Circular linked list 8

2.2.1 Insert a new node at the beginning

2.2.2 Insert a new node at the end

2.2.3 Delete the first node

2.2.4 Delete the last node

2.3 Doubly linked list 9

2.3.1 Insert a new node at the beginning

2.3.2 Insert a new node at the end

2.3.3 Insert a new node after a given node

2.3.4 Insert a new node before a given node

2.3.5 Delete the first node

2.3.6 Delete the last node

2.3.7 Delete a node after a given node

2.3.8 Delete a node before a given

CHAPTER 3 Program 13

3.1 Linear linked list 13

Output 18

3.2 Circular linked list 22

Output 25

3.3 Doubly linked list 28

Output 33

CHAPTER 4 Conclusion 37

# 1 Introduction

# Introduction to Linked Lists

Linked List is a very commonly used linear data structure which consists of group of **nodes** in a sequence.

Each node holds its own **data** and the **address of the next node** hence forming a chain like structure.

**START**

DATA

ADDRESS

DATA

ADDRESS

DATA

ADDRESS

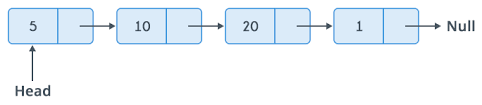
## **Types of Linked Lists**

## There are 3 different implementations of Linked List available, they are:

1. Linear Linked List
2. Doubly Linked List
3. Circular Linked List

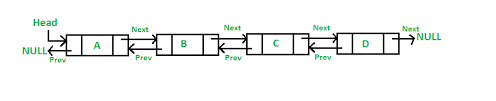
**1.Linear Linked List**

* 1. **Linear linked list** is a basic **linked list** type. **Linear linked list** is a collection of nodes **linked** together in a sequential way where each node of **linear linked list** contains a data field and an address field which contains the reference of the next node.



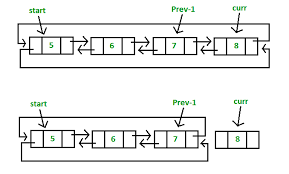
**1.2 Doubly Linked List**

**Doubly linked list** is a **linked** data structure that consists of a set of sequentially **linked** records called nodes. Each node contains three fields: two **link** fields (references to the previous and to the next node in the sequence of nodes) and one data field.



**1.3 Circular Linked List**

**Circular linked list** is a **linked list** where all nodes are connected to form a circle. There is no NULL at the end. A **circular linked list** can be a singly **circular linked list** or doubly **circular linked list**. ... We can maintain a pointer to the last inserted node and front can always be obtained as next of last.



**2 Algorithm**

**2.1 Linear linked list**

**Algorithm for traversing a linked list**

Step 1: [INITIALIZE] SET PTR=START

Step 2: Repeat Steps3 and 4 while PTR != NULL

Step 3: Apply Process to PTR ->DATA

Step 4: SET PTR=PTR-> NEXT

[END OF LOOP]

Step 5: EXIT

**Algorithm to print the number of nodes in a linked list**

Step 1: [INITIALIZE] SET COUNT=0

Step 2: [INITIALIZE] SET PTR=START

Step 3: Repeat Steps 4 and 5 while PTR != NULL

Step 4: SET COUNT=COUNT + 1

Step 5: SET PTR=PTR ->NEXT

[END OF LOOP]

Step 6: Write COUNT

Step 7: EXIT

**Algorithm to insert a new node at the beginning**

Step 1: IF AVAIL = NULL

Write OVERFLOW

Go to Step 7

[END OF IF]

Step 2: SET NEW\_NODE = AVAIL

Step 3: SET AVAIL = AVAIL-> NEXT

Step 4: SET NEW\_NODE -> DATA = VAL

Step 5: SET NEW\_NODE- >NEXT = START

Step 6: SET START = NEW\_NODE

Step 7: EXIT

**Algorithm to insert a new node at the end**

Step 1: IF AVAIL = NULL

Write OVERFLOW

Go to Step 1

[END OF IF]

Step 2: SET NEW\_NODE= AVAIL

Step 3: SET AVAIL = AVAIL-> NEXT

Step 4: SET NEW\_NODE-> DATA = VAL

Step 5: SET NEW\_NODE ->NEXT= NULL

Step 6: SET PTR = START

Step 7: Repeat Step 8 while PTR-> NEXT != NULL

Step 8: SET PTR = PTR-> NEXT

[END OF LOOP]

Step 9: SET PTR ->NEXT =NEW\_NODE

Step 10: EXIT

**Algorithm to insert a new node after a node that has value num**

Step 1: IF AVAIL = NULL

Write OVERFLOW

Go to Step 12

[END OF IF]

Step 2: SETNEW\_NODE = AVAIL

Step 3: SET AVAIL = AVAIL-> NEXT

Step 4: SETNEW\_NODE-> DATA = VAL

Step 5: SET PTR = START

Step 6: SET PREPTR = PTR

Step 7: Repeat Steps 8 and 9 while PREPTR-> DATA!=NUM

Step 8: SET PREPTR = PTR

Step 9: SET PTR = PTR-> NEXT

[END OF LOOP]

Step 10 : PREPTR-> NEXT =NEW\_NODE

Step 11: SET NEW\_NODE NEXT = PTR

Step 12: EXIT

**Algorithm to insert a new node before a node that has value NUM**

Step 1: IF AVAIL = NULL

Write OVERFLOW

Go to Step 12

[END OF IF]

Step 2: SET NEW\_NODE = AVAIL

Step 3: SET AVAIL = AVAIL->NEXT

Step 4: SETNEW\_NODE-> DATA = VAL

Step 5: SET PTR = START

Step 6: SET PREPTR = PTR

Step 7: Repeat Steps 8 and 9 while PTR-> DATA != NUM

Step 8: SET PREPTR = PTR

Step 9: SET PTR = PTR-> NEXT

[END OF LOOP]

Step 10 : PREPTR-> NEXT =NEW\_NODE

Step 11: SETNEW\_NODE-> NEXT = PTR

Step 12: EXIT

**Algorithm to delete the first node**

Step 1: IF START = NULL

Write UNDERFLOW

Go to Step 5

[END OF IF]

Step 2: SET PTR = START

Step 3: SET START = START-> NEXT

Step 4: FREE PTR

Step 5: EXIT

**Algorithm to delete the last node**

Step 1: IF START = NULL

Write UNDERFLOW

Go to Step 8

[END OF IF]

Step 2: SET PTR = START

Step 3: Repeat Steps 4 and 5 while PTR ->NEXT != NULL

Step 4: SET PREPTR = PTR

Step 5: SET PTR = PTR-> NEXT

[END OF LOOP]

Step 6: SET PREPTR-> NEXT = NULL

Step 7: FREE PTR

Step 8: EXIT

**Algorithm to delete the node after a given node**

Step 1: IF START = NULL

Write UNDERFLOW

Go to Step 1

[END OF IF]

Step 2: SET PTR = START

Step 3: SET PREPTR = PTR

Step 4: Repeat Steps 5 and 6 while PREPTR ->DATA != NUM

Step 5: SET PREPTR = PTR

Step 6: SET PTR = PTR ->NEXT

[END OF LOOP]

Step 7: SET TEMP = PTR

Step 8: SET PREPTR-> NEXT = PTR NEXT

Step 9: FREE TEMP

Step 1 : EXIT

* 1. **CIRCULAR LINKED LIST**

**Algorithm to insert a new node at the beginning**

Step 1: IF AVAIL = NULL

Write OVERFLOW

Go to Step 11

[END OF IF]

Step 2: SETNEW\_NODE = AVAIL

Step 3: SET AVAIL = AVAIL-> NEXT

Step 4: SETNEW\_NODE-> DATA = VAL

Step 5: SET PTR = START

Step 6: Repeat Step 7 while PTR->NEXT != START

Step 7: PTR = PTR ->NEXT

[END OF LOOP]

Step 8: SETNEW\_NODE-> NEXT = START

Step 9: SET PTR-> NEXT =NEW\_NODE

Step 1 : SET START =NEW\_NODE

Step 11: EXIT

**Algorithm to insert a new node at the end**

Step 1: IF AVAIL = NULL

Write OVERFLOW

Go to Step 1

[END OF IF]

Step 2: SETNEW\_NODE = AVAIL

Step 3: SET AVAIL = AVAIL-> NEXT

Step 4: SETNEW\_NODE-> DATA = VAL

Step 5: SETNEW\_NODE-> NEXT = START

Step 6: SET PTR = START

Step 7: Repeat Step 8 while PTR-> NEXT != START

Step 8: SET PTR = PTR-> NEXT

[END OF LOOP]

Step 9: SET PTR-> NEXT =NEW\_NODE

Step 10 : EXIT

**Algorithm to delete the first node**

Step 1: IF START = NULL

Write UNDERFLOW

Go to Step 8

[END OF IF]

Step 2: SET PTR = START

Step 3: Repeat Step 4 while PTR-> NEXT != START

Step 4: SET PTR = PTR-> NEXT

[END OF LOOP]

Step 5: SET PTR-> NEXT = START ->NEXT

Step 6: FREE START

Step 7: SET START = PTR-> NEXT

Step 8: EXIT

**Algorithm to delete the last node**

Step 1: IF START = NULL

Write UNDERFLOW

Go to Step 8

[END OF IF]

Step 2: SET PTR = START

Step 3: Repeat Steps 4 and 5 while PTR-> NEXT != START

Step 4: SET PREPTR = PTR

Step 5: SET PTR = PTR-> NEXT

[END OF LOOP]

Step 6: SET PREPTR ->NEXT = START

Step 7: FREE PTR

Step 8: EXIT

**2.3 DOUBLY LINKED LISTS**

**Algorithm to insert a new node at the beginning**

Step 1: IF AVAIL = NULL

Write OVERFLOW

Go to Step 9

[END OF IF]

Step 2: SETNEW\_NODE = AVAIL

Step 3: SET AVAIL = AVAIL-> NEXT

Step 4: SET NEW\_NODE-> DATA = VAL

Step 5: SET  NEW\_NODE->PREV = NULL

Step 6: SET NEW\_NODE->NEXT= START

Step 7: SET START->PREV=NEW\_NODE

Step 8: SET START =NEW\_NODE

Step 9: EXIT

**Algorithm to insert a new node at the end**

Step 1: IF AVAIL = NULL

Write OVERFLOW

Go to Step 11

[END OF IF]

Step 2: SET NEW\_NODE = AVAIL

Step 3: SET AVAIL = AVAIL ->NEXT

Step 4: SET NEW\_NODE-> DATA = VAL

Step 5: SET NEW\_NODE->NEXT= NULL

Step 6: SET PTR = START

Step 7: Repeat Step 8 while PTR-> NEXT != NULL

Step 8: SET PTR = PTR ->NEXT

[END OF LOOP]

Step 9: SET PTR NEXT =NEW\_NODE

Step 10 : SET NEW\_NODE-> PREV = PTR

Step 11: EXIT

**Algorithm to insert a new node after a given node**

Step 1: IF AVAIL = NULL

Write OVERFLOW

Go to Step 12

[END OF IF]

Step 2: SET  NEW\_NODE= AVAIL

Step 3: SET AVAIL = AVAIL ->NEXT

Step 4: SET NEW\_NODE-> DATA = VAL

Step 5: SET PTR = START

Step 6: Repeat Step 7 while PTR-> DATA != NUM

Step 7: SET  PTR = PTR->NEXT

[END OF LOOP]

Step 8: SET NEW\_NODE->NEXT = PTR-> NEXT

Step 9: SET NEW\_NODE->PREV = PTR

Step 10 : SET PTR-> NEXT =NEW\_NODE

Step 11: SET PTR-> NEXT-> PREV=NEW\_NODE

Step 12: EXIT

**Algorithm to insert a new node before a given node**

Step 1: IF AVAIL = NULL

Write OVERFLOW

Go to Step 12

[END OF IF]

Step 2: SET  NEW\_NODE = AVAIL

Step 3: SET AVAIL = AVAIL-> NEXT

Step 4: SET NEW\_NODE-> DATA = VAL

Step 5: SET  PTR = START

Step 6: Repeat Step 7 while PTR-> DATA != NUM

Step 7: SET PTR = PTR ->NEXT

[END OF LOOP]

Step 8: SET NEW\_NODE-> NEXT = PTR

Step 9: SET NEW\_NODE-> PREV = PTR ->PREV

Step 10 : SET PTR-> PREV =NEW\_NODE

Step 11:  SET PTR->PREV-> NEXT=NEW\_NODE

Step 12:EXIT

**Algorithm to delete the first node**

Step 1: IF START = NULL

Write UNDERFLOW

Go to Step 6

[END OF IF]

Step 2: SET PTR = START

Step 3: SET START = START ->NEXT

Step 4: SET START ->PREV = NULL

Step 5: FREE PTR

Step 6: EXIT

**Algorithm to delete the last node**

Step 1: IF START = NULL

Write UNDERFLOW

Go to Step 7

[END OF IF]

Step 2: SET  PTR = START

[END OF LOOP]

Step 3: Repeat Step 4 while PTR-> NEXT != NULL

Step 4: SET PTR = PTR-> NEXT

Step 5: SET PTR-> PREV-> NEXT = NULL

Step 6: FREE PTR

Step 7: EXIT

**Algorithm to delete a node after a given node**

Step 1: IF START = NULL

Write UNDERFLOW

Go to Step 9

[END OF IF]

Step 2: SET  PTR = START

[END OF LOOP]

Step 3: Repeat Step 4 while PTR-> DATA != NUM

Step 4: SET PTR = PTR-> NEXT

Step 5: SET TEMP = PTR-> NEXT

Step 6: SET  PTR -> NEXT = TEMP-> NEXT

Step 7: SET TEMP-> NEXT-> PREV = PTR

Step 8: FREE TEMP

Step 9: EXIT

**Algorithm to delete a node before a given**

Step 1: IF START = NULL

Write UNDERFLOW

Go to Step 9

[END OF IF]

Step 2: SET  PTR = START

[END OF LOOP]

Step 3: Repeat Step 4 while PTR-> DATA != NUM

Step 4: SET PTR = PTR-> NEXT

Step 5: SET TEMP = PTR-> PREV

Step 6: SET  TEMP-> PREV-> NEXT = PTR

Step 7: SET PTR-> PREV = TEMP-> PREV

Step 8: FREE TEMP

Step 9: EXIT

**3 Program**

**3.1 Write a program of linear linked list to perform the operation of insertion and deletion**.

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <malloc.h>

struct node

{

int data;

struct node \*next;

};

struct node \*start = NULL;

struct node \*create\_ll(struct node \*);

struct node \*display(struct node \*);

struct node \*insert\_beg(struct node \*);

struct node \*insert\_end(struct node \*);

struct node \*insert\_before(struct node \*);

struct node \*insert\_after(struct node \*);

struct node \*delete\_beg(struct node \*);

struct node \*delete\_end(struct node \*);

struct node \*delete\_node(struct node \*);

struct node \*delete\_after(struct node \*);

struct node \*delete\_list(struct node \*);

struct node \*sort\_list(struct node \*);

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

{

int option;

do

{

printf("\n\n \*\*\*\*\*MAIN MENU \*\*\*\*\*");

printf("\n 1: Create a list");

printf("\n 2: Display the list");

printf("\n 3: Add a node at the beginning");

printf("\n 4: Add a node at the end");

printf("\n 5: Add a node before a given node");

printf("\n 6: Add a node after a given node");

printf("\n 7: Delete a node from the beginning");

printf("\n 8: Delete a node from the end");

printf("\n 9: Delete a given node");

printf("\n 10: Delete a node after a given node");

printf("\n 11: Delete the entire list");

printf("\n 12: Sort the list");

printf("\n 13: EXIT");

printf("\n\n Enter your option : ");

scanf("%d", &option);

switch(option)

{

case 1:start = create\_ll(start);

printf("\n LINKED LIST CREATED");

break;

case 2: start = display(start);

break;

case 3: start = insert\_beg(start);

break;

case 4: start = insert\_end(start);

break;

case 5: start = insert\_before(start);

break;

case 6: start = insert\_after(start);

break;

case 7: start = delete\_beg(start);

break;

case 8: start = delete\_end(start);

break;

case 9: start = delete\_node(start);

break;

case 10: start = delete\_after(start);

break;

case 11: start = delete\_list(start);

printf("\n LINKED LIST DELETED");

break;

case 12: start = sort\_list(start);

break;

}

}

while(option !=13);

getch();

return 0;

}

struct node \*create\_ll(struct node \*start)

{

struct node \*new\_node, \*ptr;

int num;

printf("\n Enter -1 to end");

printf("\n Enter the data : ");

scanf("%d", &num);

while(num!=-1)

{

new\_node = (struct node\*)malloc(sizeof(struct node));

new\_node -> data=num;

if(start==NULL)

{

new\_node -> next = NULL;

start = new\_node;

}

else

{

ptr=start;

while(ptr->next!=NULL)

ptr=ptr->next;

ptr->next = new\_node;

new\_node->next=NULL;

}

printf("\n Enter the data : ");

scanf("%d", &num);

}

return start;

}

struct node \*display(struct node \*start)

{

struct node \*ptr;

ptr = start;

while(ptr != NULL)

{

printf("\t %d", ptr -> data);

ptr = ptr -> next;

}

return start;

}

struct node \*insert\_beg(struct node \*start)

{

struct node \*new\_node;

int num;

printf("\n Enter the data : ");

scanf("%d", &num);

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node -> data = num;

new\_node -> next = start;

start = new\_node; return start;

}

struct node \*insert\_end(struct node \*start)

{

struct node \*ptr, \*new\_node;

int num;

printf("\n Enter the data : ");

scanf("%d", &num);

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node -> data = num;

new\_node -> next = NULL;

ptr = start;

while(ptr -> next != NULL)

ptr = ptr -> next;

ptr -> next = new\_node;

return start;

}

struct node \*insert\_before(struct node \*start)

{

struct node \*new\_node, \*ptr, \*preptr; int num, val;

printf("\n Enter the data : ");

scanf("%d", &num);

printf("\n Enter the value before which the data has to be inserted : ");

scanf("%d", &val);

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node -> data = num;

ptr = start;

while(ptr -> data != val)

{

preptr = ptr; ptr = ptr -> next;

}

preptr -> next = new\_node; new\_node -> next = ptr;

return start;

}

struct node \*insert\_after(struct node \*start)

{

struct node \*new\_node, \*ptr, \*preptr;

int num, val;

printf("\n Enter the data : ");

scanf("%d", &num);

printf("\n Enter the value after which the data has to be inserted : ");

scanf("%d", &val);

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node -> data = num;

ptr = start;

preptr = ptr;

while(preptr -> data != val)

{

preptr = ptr;

ptr = ptr -> next;

}

preptr -> next=new\_node;

new\_node -> next = ptr;

return start;

}

struct node \*delete\_beg(struct node \*start)

{

struct node \*ptr;

ptr = start;

start = start -> next;

free(ptr); return start;

}

struct node \*delete\_end(struct node \*start)

{

struct node \*ptr, \*preptr; ptr = start;

while(ptr -> next != NULL)

{

preptr = ptr;

ptr = ptr -> next;

}

preptr -> next = NULL;

free(ptr); return start;

}

struct node \*delete\_node(struct node \*start)

{

struct node \*ptr, \*preptr; int val;

printf("\n Enter the value of the node which has to be deleted : ");

scanf("%d", &val);

ptr = start;

if(ptr -> data == val)

{

start = delete\_beg(start);

return start;

}

else

{

while(ptr -> data != val)

{

preptr = ptr;

ptr = ptr -> next;

}

preptr -> next = ptr -> next;

free(ptr);

return start;

}

}

struct node \*delete\_after(struct node \*start)

{

struct node \*ptr, \*preptr;

int val;

printf("\n Enter the value after which the node has to deleted : ");

scanf("%d", &val);

ptr = start; preptr = ptr;

while(preptr -> data != val)

{

preptr = ptr;

ptr = ptr -> next;

}

preptr -> next=ptr -> next;

free(ptr);

return start;

}

struct node \*delete\_list(struct node \*start)

{

struct node \*ptr;

return start;

}

struct node \*sort\_list(struct node \*start)

{

struct node \*ptr1, \*ptr2;

int temp;

ptr1 = start;

while(ptr1 -> next != NULL)

{

ptr2 = ptr1 -> next;

while(ptr2 != NULL)

{

if(ptr1 -> data > ptr2 -> data)

{

temp = ptr1 -> data;

ptr1 -> data = ptr2 -> data;

ptr2 -> data = temp;

}

ptr2 = ptr2 -> next;

}

ptr1 = ptr1 -> next;

}

return start;

}

**Output**

\*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 1** Enter -1 to end Enter the data : 45 Enter the data : 85 Enter the data : -1 **LINKED LIST CREATED** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 2 45 85** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 3 Enter the data : 25** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 4 Enter the data : 65** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 2 25 45 85 65** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 5 Enter the data : 95 Enter the value before which the data has to be inserted : 85** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 2 25 45 95 85 65** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 7** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 8**  \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 2 45 95 85** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 9 Enter the value of the node which has to be deleted : 95** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 2 45 85**  \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 11 LINKED LIST DELETED** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a given node 10: Delete a node after a given node 11: Delete the entire list 12: Sort the list 13: EXIT **Enter your option : 13** Process returned 0 (0x0) execution time : 300.464 ss

**3.2** **Write a program of circular linked list to perform the operation of insertion and deletion.**

#include <stdio.h>

#include <conio.h>

#include <malloc.h>

struct node { int data; struct node \*next; };

struct node \*start = NULL;

struct node \*create\_cll(struct node \*);

struct node \*display(struct node \*);

struct node \*insert\_beg(struct node \*);

struct node \*insert\_end(struct node \*);

struct node \*delete\_beg(struct node \*);

struct node \*delete\_end(struct node \*);

struct node \*delete\_after(struct node \*);

struct node \*delete\_list(struct node \*);

int main()

{

int option;

int clrscr();

do

{

printf("\n\n \*\*\*\*\*MAIN MENU \*\*\*\*\*");

printf("\n 1: Create a list");

printf("\n 2: Display the list");

printf("\n 3: Add a node at the beginning");

printf("\n 4: Add a node at the end");

printf("\n 5: Delete a node from the beginning");

printf("\n 6: Delete a node from the end");

printf("\n 7: Delete a node after a given node");

printf("\n 8: Delete the entire list");

printf("\n 9: EXIT");

printf("\n\n Enter your option : ");

scanf("%d", &option);

switch(option)

{

case 1: printf("enter -1 to end the list");

start = create\_cll(start);

printf("\n CIRCULAR LINKED LIST CREATED");

break;

case 2: start = display(start);

break;

case 3: start = insert\_beg(start);

break;

case 4: start = insert\_end(start);

break;

case 5: start = delete\_beg(start);

break;

case 6: start = delete\_end(start);

break;

case 7: start = delete\_after(start);

break;

case 8: start = delete\_list(start);

printf("\n CIRCULAR LINKED LIST DELETED");

break;

}

}

while(option !=9);

getch();

return 0;

}

struct node \*create\_cll(struct node \*start)

{

struct node \*new\_node, \*ptr;

int num;

printf("\n Enter the data : ");

scanf("%d", &num);

while(num!=-1)

{

new\_node = (struct node\*)malloc(sizeof(struct node));

new\_node -> data = num;

if(start == NULL)

{

new\_node -> next = new\_node;

start = new\_node;

}

else

{

ptr = start;

while(ptr -> next != start)

ptr = ptr -> next;

ptr -> next = new\_node;

new\_node -> next = start;

}

printf("\n Enter the data : ");

scanf("%d", &num);

}

return start;

}

struct node \*display(struct node \*start)

{

struct node \*ptr; ptr=start;

while(ptr -> next != start)

{

printf("\t %d", ptr -> data);

ptr = ptr -> next;

}

printf("\t %d", ptr -> data);

return start;

}

struct node \*insert\_beg(struct node \*start)

{

struct node \*new\_node, \*ptr;

int num;

printf("\n Enter the data : ");

scanf("%d", &num);

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node -> data = num;

ptr = start;

while(ptr -> next != start)

ptr = ptr -> next;

ptr -> next = new\_node;

new\_node -> next = start;

start = new\_node;

return start;

}

struct node \*insert\_end(struct node \*start)

{

struct node \*ptr, \*new\_node;

int num;

printf("\n Enter the data : ");

scanf("%d", &num);

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node -> data = num;

ptr = start;

while(ptr -> next != start)

ptr = ptr -> next;

ptr -> next = new\_node;

new\_node -> next = start;

return start;

}

struct node \*delete\_beg(struct node \*start)

{

struct node \*ptr;

ptr = start;

while(ptr -> next != start)

ptr = ptr -> next;

ptr -> next = start -> next;

free(start);

start = ptr -> next;

return start;

}

struct node \*delete\_end(struct node \*start)

{

struct node \*ptr, \*preptr;

ptr = start;

while(ptr -> next != start)

{

preptr = ptr;

ptr = ptr -> next;

}

preptr -> next = ptr -> next;

free(ptr);

return start;

}

struct node \*delete\_after(struct node \*start)

{

struct node \*ptr, \*preptr;

int val;

printf("\n Enter the value after which the node has to deleted : ");

scanf("%d", &val);

ptr = start;

preptr = ptr;

while(preptr -> data != val)

{

preptr = ptr;

ptr = ptr -> next;

}

preptr -> next = ptr -> next;

if(ptr == start)

start = preptr -> next;

free(ptr);

return start;

}

struct node \*delete\_list(struct node \*start)

{

struct node \*ptr;

ptr = start;

while(ptr -> next != start)

start = delete\_end(start);

free(start);

return start;

}

**Output**

\*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Delete a node from the beginning 6: Delete a node from the end 7: Delete a node after a given node 8: Delete the entire list 9: EXIT **Enter your option : 01 enter -1 to end the list Enter the data : 21 Enter the data : 23 Enter the data : -1**  **CIRCULAR LINKED LIST CREATED** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Delete a node from the beginning 6: Delete a node from the end 7: Delete a node after a given node 8: Delete the entire list 9: EXIT **Enter your option : 2 21 23** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Delete a node from the beginning 6: Delete a node from the end 7: Delete a node after a given node 8: Delete the entire list 9: EXIT **Enter your option : 3 Enter the data : 85** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Delete a node from the beginning 6: Delete a node from the end 7: Delete a node after a given node 8: Delete the entire list 9: EXIT **Enter your option : 2 85 21 23** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Delete a node from the beginning 6: Delete a node from the end 7: Delete a node after a given node 8: Delete the entire list 9: EXIT **Enter your option : 4 Enter the data : 69** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Delete a node from the beginning 6: Delete a node from the end 7: Delete a node after a given node 8: Delete the entire list 9: EXIT **Enter your option : 2 85 21 23 69** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Delete a node from the beginning 6: Delete a node from the end 7: Delete a node after a given node 8: Delete the entire list 9: EXIT **Enter your option : 5**  \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Delete a node from the beginning 6: Delete a node from the end 7: Delete a node after a given node 8: Delete the entire list 9: EXIT **Enter your option : 2 21 23 69** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Delete a node from the beginning 6: Delete a node from the end 7: Delete a node after a given node 8: Delete the entire list 9: EXIT **Enter your option : 6** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Delete a node from the beginning 6: Delete a node from the end 7: Delete a node after a given node 8: Delete the entire list 9: EXIT **Enter your option : 2 21 23** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Delete a node from the beginning 6: Delete a node from the end 7: Delete a node after a given node 8: Delete the entire list 9: EXIT **Enter your option : 7 Enter the value after which the node has to deleted : 21** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Delete a node from the beginning 6: Delete a node from the end 7: Delete a node after a given node 8: Delete the entire list 9: EXIT **Enter your option : 8 CIRCULAR LINKED LIST DELETED** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Delete a node from the beginning 6: Delete a node from the end 7: Delete a node after a given node 8: Delete the entire list **9: EXIT Enter your option : 9 Process returned 0 (0x0) execution time : 91.150 s**

**3.3 Write a program of Doubly linked list to perform the operation of insertion and deletion.**

#include <stdio.h>

#include <conio.h>

#include <malloc.h>

struct node

{

struct node \*next;

int data;

struct node \*prev;

};

struct node \*start = NULL;

struct node \*create\_ll(struct node \*);

struct node \*display(struct node \*);

struct node \*insert\_beg(struct node \*);

struct node \*insert\_end(struct node \*);

struct node \*insert\_before(struct node \*);

struct node \*insert\_after(struct node \*);

struct node \*delete\_beg(struct node \*);

struct node \*delete\_end(struct node \*);

struct node \*delete\_before(struct node \*);

struct node \*delete\_after(struct node \*);

struct node \*delete\_list(struct node \*);

int main()

{

int option;

int clrscr();

do

{

printf("\n\n \*\*\*\*\*MAIN MENU \*\*\*\*\*");

printf("\n 1: Create a list");

printf("\n 2: Display the list");

printf("\n 3: Add a node at the beginning");

printf("\n 4: Add a node at the end");

printf("\n 5: Add a node before a given node");

printf("\n 6: Add a node after a given node");

printf("\n 7: Delete a node from the beginning");

printf("\n 8: Delete a node from the end");

printf("\n 9: Delete a node before a given node");

printf("\n 10: Delete a node after a given node");

printf("\n 11: Delete the entire list");

printf("\n 12: EXIT");

printf("\n\n Enter your option : ");

scanf("%d", &option);

switch(option)

{

case 1:printf("enter -1 to end the list");

start = create\_ll(start);

printf("\n DOUBLY LINKED LIST CREATED");

break;

case 2: start = display(start);

break;

case 3: start = insert\_beg(start);

break;

case 4: start = insert\_end(start);

break;

case 5: start = insert\_before(start);

break;

case 6: start = insert\_after(start);

break;

case 7: start = delete\_beg(start);

break;

case 8: start = delete\_end(start);

break;

case 9: start = delete\_before(start);

break;

case 10: start = delete\_after(start);

break;

case 11: start = delete\_list(start);

printf("\n DOUBLY LINKED LIST DELETED");

break;

}

}

while(option != 12);

getch();

return 0;

}

struct node \*create\_ll(struct node \*start)

{

struct node \*new\_node, \*ptr;

int num;

printf("\n Enter the data : ");

scanf("%d", &num);

while(num != -1)

{

if(start == NULL)

{

new\_node = (struct node\*)malloc(sizeof(struct node));

new\_node -> prev = NULL;

new\_node -> data = num;

new\_node -> next = NULL;

start = new\_node;

}

else

{

ptr=start;

new\_node = (struct node\*)malloc(sizeof(struct node));

new\_node ->data=num;

while(ptr->next!=NULL)

ptr = ptr->next;

ptr ->next = new\_node;

new\_node ->prev=ptr;

new\_node ->next=NULL;

}

printf("\n Enter the data : ");

scanf("%d", &num);

}

return start;

}

struct node \*display(struct node \*start)

{

struct node \*ptr;

ptr=start;

while(ptr!=NULL)

{

printf("\t %d", ptr -> data);

ptr = ptr -> next;

}

return start;

}

struct node \*insert\_beg(struct node \*start)

{

struct node \*new\_node;

int num;

printf("\n Enter the data : ");

scanf("%d", &num);

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node -> data = num;

start -> prev = new\_node;

new\_node -> next = start;

new\_node -> prev = NULL;

start = new\_node;

return start;

}

struct node \*insert\_end(struct node \*start)

{

struct node \*ptr, \*new\_node;

int num;

printf("\n Enter the data : ");

scanf("%d", &num);

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node -> data = num; ptr=start;

while(ptr -> next != NULL)

ptr = ptr -> next;

ptr -> next = new\_node;

new\_node -> prev = ptr;

new\_node -> next = NULL;

return start;

}

struct node \*insert\_before(struct node \*start)

{

struct node \*new\_node, \*ptr;

int num, val;

printf("\n Enter the data : ");

scanf("%d", &num);

printf("\n Enter the value before which the data has to be inserted : ");

scanf("%d", &val);

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node -> data = num;

ptr = start;

while(ptr -> data != val)

ptr = ptr -> next;

new\_node -> next = ptr;

new\_node -> prev = ptr-> prev;

ptr -> prev -> next = new\_node;

ptr -> prev = new\_node;

return start;

}

struct node \*insert\_after(struct node \*start)

{

struct node \*new\_node, \*ptr;

int num, val;

printf("\n Enter the data : ");

scanf("%d", &num);

printf("\n Enter the value after which the data has to be inserted : ");

scanf("%d", &val);

new\_node = (struct node \*)malloc(sizeof(struct node));

new\_node -> data = num;

ptr = start;

while(ptr -> data != val) ptr = ptr -> next;

new\_node -> prev = ptr;

new\_node -> next = ptr -> next;

ptr -> next -> prev = new\_node;

ptr -> next = new\_node;

return start;

}

struct node \*delete\_beg(struct node \*start)

{

struct node \*ptr;

ptr = start; start = start -> next;

start -> prev = NULL;

free(ptr);

return start;

}

struct node \*delete\_end(struct node \*start)

{

struct node \*ptr;

ptr = start;

while(ptr -> next != NULL)

ptr = ptr -> next;

ptr -> prev -> next = NULL;

free(ptr);

return start;

}

struct node \*delete\_after(struct node \*start)

{

struct node \*ptr, \*temp; int val;

printf("\n Enter the value after which the node has to deleted : ");

scanf("%d", &val);

ptr = start;

while(ptr -> data != val)

ptr = ptr -> next;

temp = ptr -> next;

ptr -> next = temp -> next;

temp -> next -> prev = ptr;

free(temp);

return start;

}

struct node \*delete\_before(struct node \*start)

{

struct node \*ptr, \*temp;

int val;

printf("\n Enter the value before which the node has to deleted : ");

scanf("%d", &val);

ptr = start;

while(ptr -> data != val)

ptr = ptr -> next;

temp = ptr -> prev;

if(temp == start)

start = delete\_beg(start);

else

{

ptr -> prev = temp -> prev;

temp -> prev -> next = ptr;

}

free(temp);

return start;

}

struct node \*delete\_list(struct node \*start)

{

while(start != NULL)

start = delete\_beg(start);

return start;

}

**Output**

\*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a node before a given node 10: Delete a node after a given node 11: Delete the entire list 12: EXIT **Enter your option : 1 enter -1 to end the list Enter the data : 56 Enter the data : 54 Enter the data : -1 DOUBLY LINKED LIST CREATED** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a node before a given node 10: Delete a node after a given node 11: Delete the entire list 12: EXIT **Enter your option : 2 56 54** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a node before a given node 10: Delete a node after a given node 11: Delete the entire list 12: EXIT **Enter your option : 3 Enter the data : 89** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a node before a given node 10: Delete a node after a given node 11: Delete the entire list 12: EXIT **Enter your option : 4 Enter the data : 87** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a node before a given node 10: Delete a node after a given node 11: Delete the entire list 12: EXIT **Enter your option : 2 89 56 54 87** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a node before a given node 10: Delete a node after a given node 11: Delete the entire list 12: EXIT **Enter your option : 6 Enter the data : 52 Enter the value after which the data has to be inserted : 54** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a node before a given node 10: Delete a node after a given node 11: Delete the entire list 12: EXIT **Enter your option : 2 89 56 54 52 87** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a node before a given node 10: Delete a node after a given node 11: Delete the entire list 12: EXIT **Enter your option : 7** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a node before a given node 10: Delete a node after a given node 11: Delete the entire list 12: EXIT **Enter your option : 8** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a node before a given node 10: Delete a node after a given node 11: Delete the entire list 12: EXIT **Enter your option : 2 56 54 52** \*\*\*\*\*MAIN MENU \*\*\*\*\* 1: Create a list 2: Display the list 3: Add a node at the beginning 4: Add a node at the end 5: Add a node before a given node 6: Add a node after a given node 7: Delete a node from the beginning 8: Delete a node from the end 9: Delete a node before a given node 10: Delete a node after a given node 11: Delete the entire list **12: EXIT Enter your option : 12 Process returned 0 (0x0) execution time : 120.164 s**