Department Of Computer Science

Gujarat University

Certificate

Roll No: ___40___ *Seat No:* _______

This is to certify that Mr. / ~~Ms~~. **Akshit Trivedi Ajaybhai** *student of MCA Semester - I, has duly completed his/~~her~~ term work for the semester ending in February 2022, in the subject of* **Data Structures** *towards partial fulfillment of his / ~~her~~ Degree of Masters in Computer Science & Application.*

28/02/2022

Date of Submission *Internal Faculty*

Head of Department

Department Of Computer Science
Gujarat University

MCA – 1

Subject: -____Data Structures________________________

Name: -___________Akshit Trivedi Ajaybhai_______________

Roll No.: -___40____ **Exam Seat No.: -________**

Sr. No.	**Contents**	**Pg. No**	**Date**	**Signature**
1	Operation on Singly Linked List	4	05/12/2021	
2	Operation on Circular Linked List	24	12/12/2021	
3	Operation on Doubly Linked List	34	30/12/2021	
4	Operation on Stack using Array	52	05/01/2022	
5	Operation on Stack using Linked List	57	05/01/2022	
6	Operation on Queue using Array	62	19/01/2022	
7	Operation on Queue using Linked List	67	19/01/2022	
8	Operation on Circular Queue using Array	72	24/01/2022	
9	Operation on Circular Queue using Linked List	77	24/01/2022	
10	Operation on Priority Queue using Array	82	30/01/2022	
11	Operation on Priority Queue using Linked List	89	30/01/2022	
12	Operation on Binary Search Tree	95	06/02/2022	
13	Operation on Postfix Evaluation	104	08/02/2022	

Rollwala Computer Center

DATA STRUCTURES

Assignment

05 Feb, 2022

Name: **Akshit Trivedi**

Roll No: **40**

Course: **Master of Computer Application**

Sem: **1**

DS ASSIGNMENT

Roll No: 40 **Name: Akshit Trivedi**
Class: MCA-1 **Year: 2021-22**

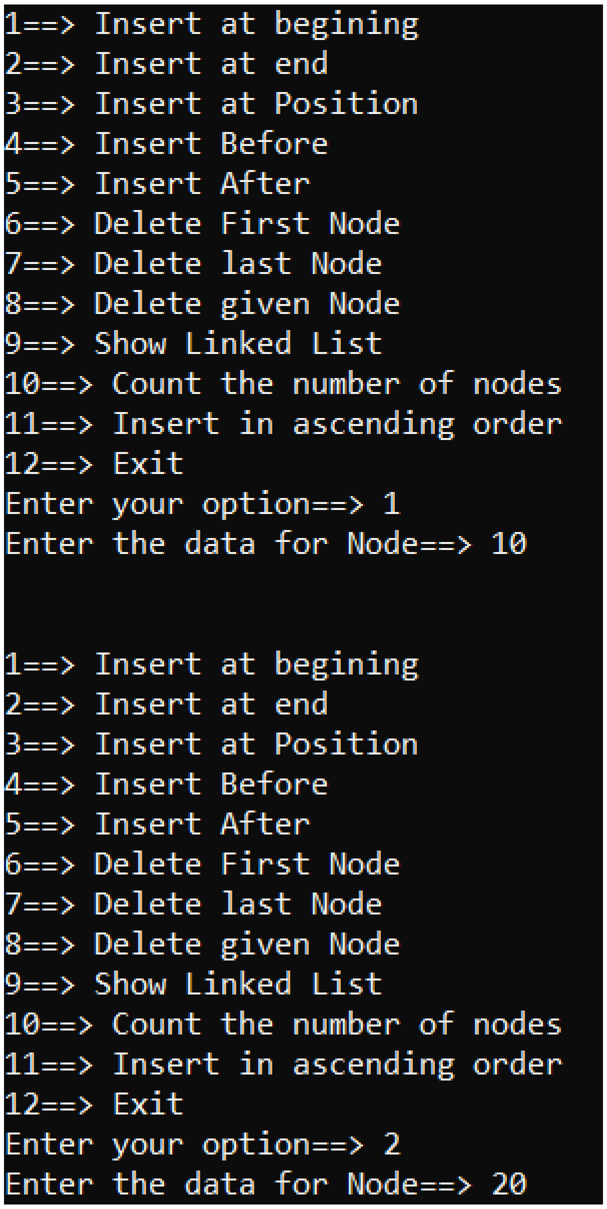
1. Singly Linked List.

INPUT:

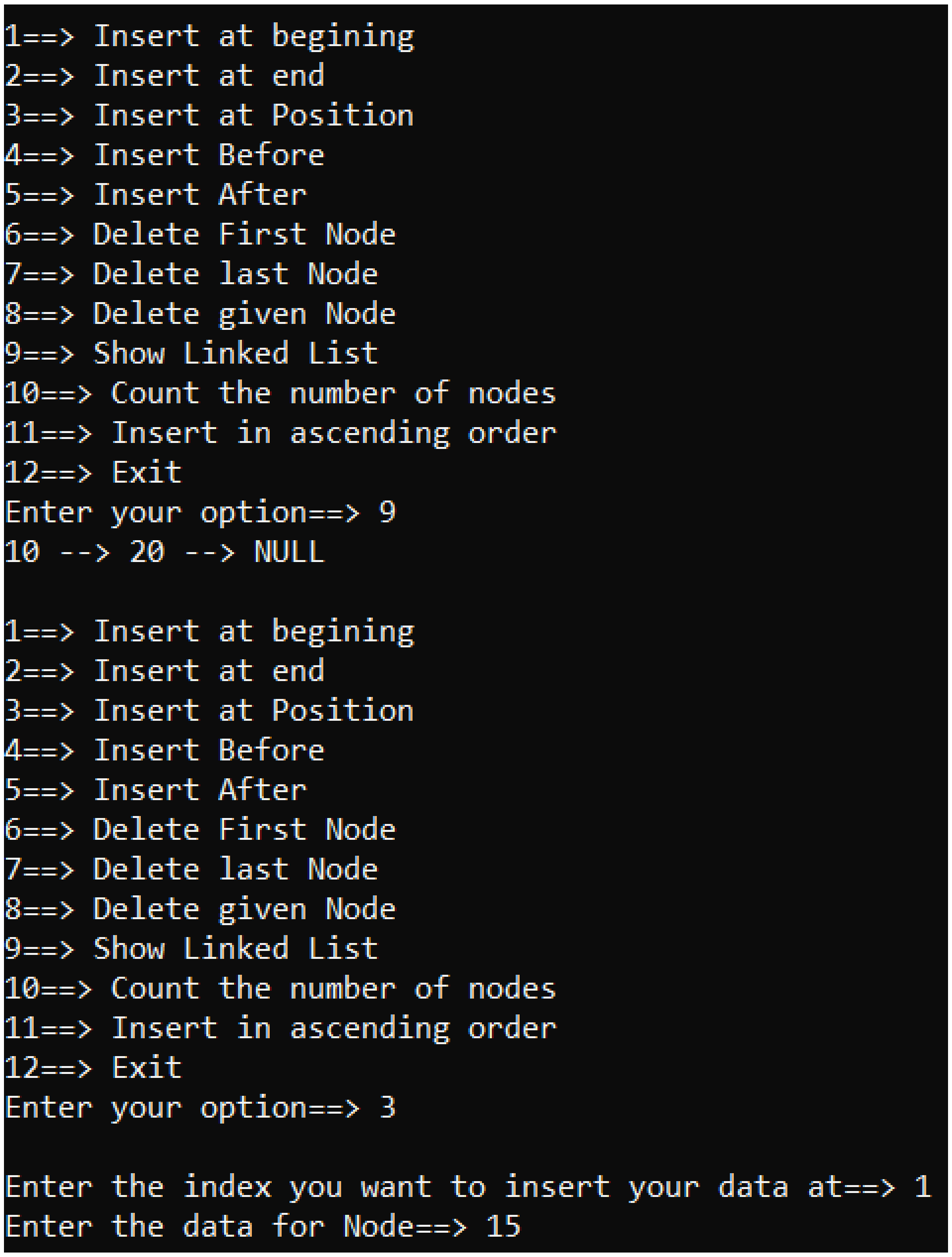
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

struct node
{
    int data;
    struct node *next;
}*new_node, *temp, *head, *tail;


void create()
{
    int data;
    new_node = (struct node*)malloc(sizeof(struct node));
    printf("Enter the data for Node==> ");
    scanf("%d",&data);
    new_node->data = data;
    new_node->next=NULL;
    head = new_node;
    tail = new_node;
}

void insert_at_begin()
{
    int data;
    if(head==NULL)
    {
        create();
    }
    else
    {
        new_node = (struct node*)malloc(sizeof(struct node));
        printf("Enter the data for Node==> ");
        scanf("%d",&data);
        new_node->data = data;
        new_node->next = head;
        head = new_node;
    }
}

void insert_at_end()
{
    if(head==NULL)
    {
        create();
    }
    else
    {
        int data;
        temp = head;
        while(temp->next != NULL)
        {
            temp = temp->next;
        }
        new_node = (struct node*)malloc(sizeof(struct node));
        printf("Enter the data for Node==> ");
        scanf("%d",&data);
        new_node->data = data;
        new_node->next = NULL;
        temp->next = new_node;
    }
}

struct node* node_at_index(int index)
{
    int count = 1;
    temp = head;
    if(index!=0)
    {
        while (count<=index & temp->next!=NULL)
        {
            temp=temp->next;
            count++;
        }
        if(temp->next==NULL) return NULL;
    }
    return temp;
}

void insert_at_index()
{
    int index,data;
    if(head==NULL)
    {
        printf("Linked list is null.");
        create();
        return;
    }
    printf("\nEnter the index you want to insert your data at==> ");
    scanf("%d", &index);
    if(index==0)
    {
        insert_at_begin();
        return;
    }
    temp = node_at_index(index-1);
    if(temp==NULL)
    {
        printf("This index doesn't exist");
        return;
    }

    new_node = (struct node*)malloc(sizeof(struct node));
    printf("Enter the data for Node==> ");
    scanf("%d", &data);
    new_node->data = data;
    if(temp->next==NULL)
    {
        new_node->next = NULL;
    }
    else
    {
        new_node->next = temp->next;
    }
    temp->next=new_node;
}

void insert_before()
{
    int val_search, data, flag=0;
    struct node *ptr_prev, *ptr;
    new_node = (struct node*)malloc(sizeof(struct node));
    ptr_prev = head;
    ptr = head;
    printf("Enter the value you want to insert before==> ");
    scanf("%d",&val_search);

    if(val_search==ptr->data)
    {
        insert_at_begin();
        return;
    }

    while(ptr!=NULL)
    {
        if(ptr->data==val_search)
        {
            flag=1;
            break;
        }
        ptr_prev = ptr;
        ptr = ptr->next;
    }

    if(flag==0)
    {
        printf("%d does not exist in the linked list", val_search);
        return;
    }

    printf("Enter the data==> ");
    scanf("%d", &data);
```

```c
    new_node->data = data;
    ptr_prev->next = new_node;
    new_node->next = ptr;
}

void insert_after()
{
    int val_search, data, flag=0;
    struct node *ptr;
    new_node = (struct node*)malloc(sizeof(struct node));
    ptr = head;
    printf("Enter the value you want to insert after==> ");
    scanf("%d",&val_search);

    while(ptr!=NULL)
    {
        if(ptr->data==val_search)
        {
            flag=1;
            break;
        }
        ptr = ptr->next;
    }

    if(flag==0)
    {
        printf("%d does not exist in the linked list", val_search);
        return;
    }

    printf("Enter the data==> ");
    scanf("%d", &data);
    new_node->data = data;
    new_node->next = ptr->next;
    ptr->next = new_node;
}

void del_first()
{
    if(head==NULL)
    {
        printf("Linked list is empty");
        return;
    }
    temp = head->next;
    free(head);
    head = temp;
}

void del_last()
{
    struct node *preptr, *ptr;
    if(head == NULL)
    {
        printf("LInked list is empty");
        return;
    }
    if(head->next==NULL)
    {
        free(head);
        head=NULL;
        return;
    }

    ptr = head;
    preptr = head;

    while(ptr->next != NULL)
    {
        preptr = ptr;
        ptr = ptr->next;
    }
    free(ptr);
    preptr->next = NULL;
}

void del_data()
{
    int data, flag=0;
    struct node *ptr, *preptr;
    if(head == NULL)
    {
        printf("Linked List is empty");
        return;
    }
    printf("Enter the data==> ");
    scanf("%d",&data);
    if(head->data == data)
    {
        del_first();
    }
    ptr = head;
    preptr = head;

    while(ptr != NULL)
    {
        if(ptr->data == data)
        {
            preptr->next = ptr->next;
            free(ptr);
            flag=1;
            break;
        }
        preptr = ptr;
        ptr = ptr->next;
    }
    if(flag==0) printf("%d doesn't exist in the linked list", data);
}

void count()
{
    int count=0;
    temp = head;
    if(head == NULL)
    {
        printf("Linked List is empty");
        return;
    }
    while(temp != NULL)
    {
        count++;
        temp = temp->next;
    }
    printf("Count==> %d", count);
}

void show()
{
    if(head==NULL)
    {
        printf("Linked List is empty");
        return;
    }
    temp = head;
    while(temp!=NULL)
    {
        printf("%d --> ", temp->data);
        temp = temp->next;
    }
    printf("NULL");
}

void insert_ascending()
{
    int data;
    struct node *ptr, *preptr;
    ptr = head;
    preptr = head;
    if(head == NULL)
    {
        create();
        return;
    }
    printf("Enter the data==> ");
    scanf("%d", &data);
    new_node = (struct node*)malloc(sizeof(struct node));
    while(ptr != NULL && ptr->data <= data )
    {
        preptr = ptr;
        ptr = ptr->next;
    }
    new_node->data = data;
    new_node->next = ptr;
    if(head->data >= data)
    {
        head = new_node;
        return;
    }
    preptr->next = new_node;
}

void main()
{
    int op=0;
    while(op!=12)
    {
        printf("\n\n1==> Insert at begining");
        printf("\n2==> Insert at end");
        printf("\n3==> Insert at Position");
        printf("\n4==> Insert Before");
        printf("\n5==> Insert After");
        printf("\n6==> Delete First Node");
        printf("\n7==> Delete last Node");
        printf("\n8==> Delete given Node");
        printf("\n9==> Show Linked List");
        printf("\n10==> Count the number of nodes");
        printf("\n11==> Insert in ascending order");
        printf("\n12==> Exit");
        printf("\nEnter your option==> ");
        scanf("%d",&op);
        switch (op)
        {
            case 1 : insert_at_begin();
                break;
            case 2 : insert_at_end();
                break;

            case 3 : insert_at_index();
                break;

            case 4 : insert_before();
                break;

            case 5 : insert_after();
                break;

            case 6 : del_first();
                break;

            case 7 : del_last();
                break;

            case 8 : del_data();
                break;

            case 9 : show();
                break;

            case 10 : count();
                break;

            case 11 : insert_ascending();
                break;

            case 12 : printf("\nExiting!!!");
                break;

            default : printf("\nInvalid Input");
                break;
        }
    }
}
```

OUTPUT:



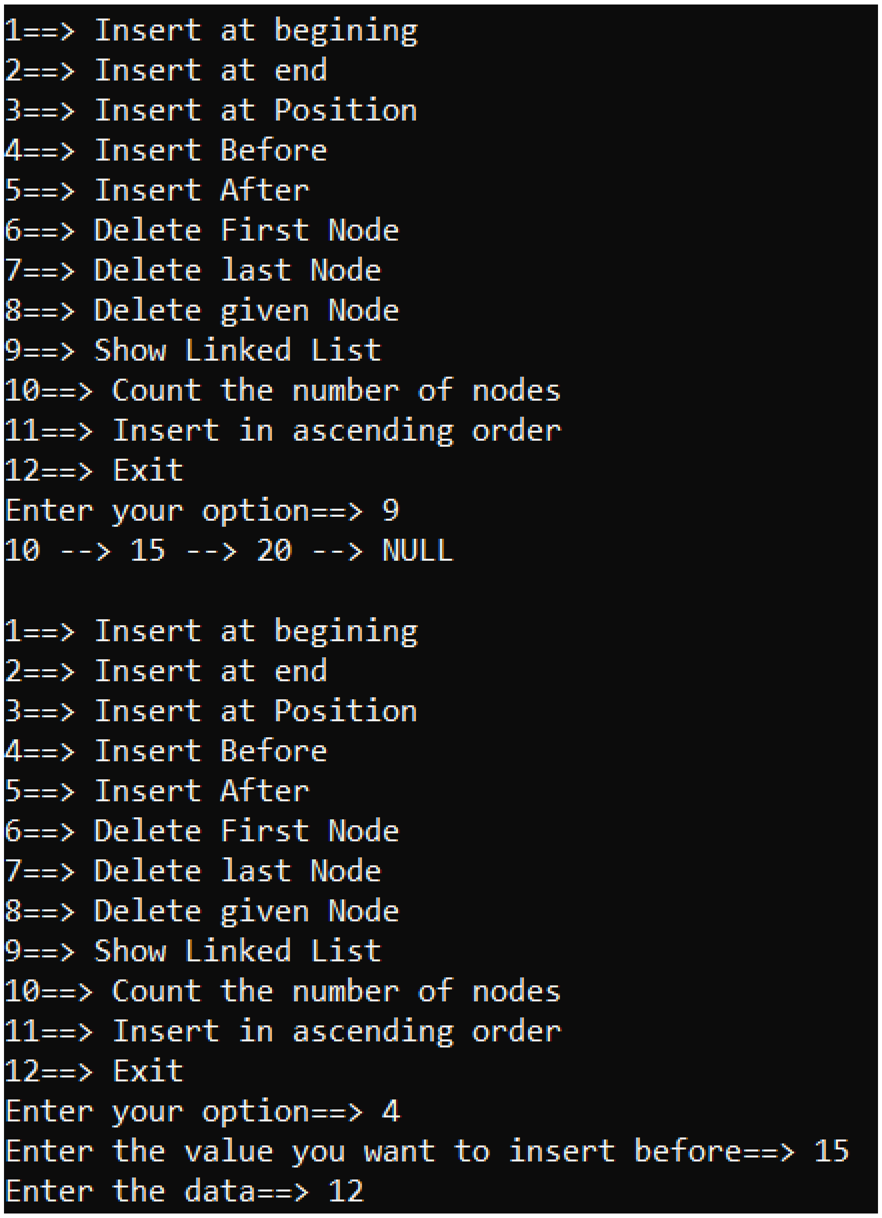
```
1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 1
Enter the data for Node==> 10


1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 2
Enter the data for Node==> 20
```



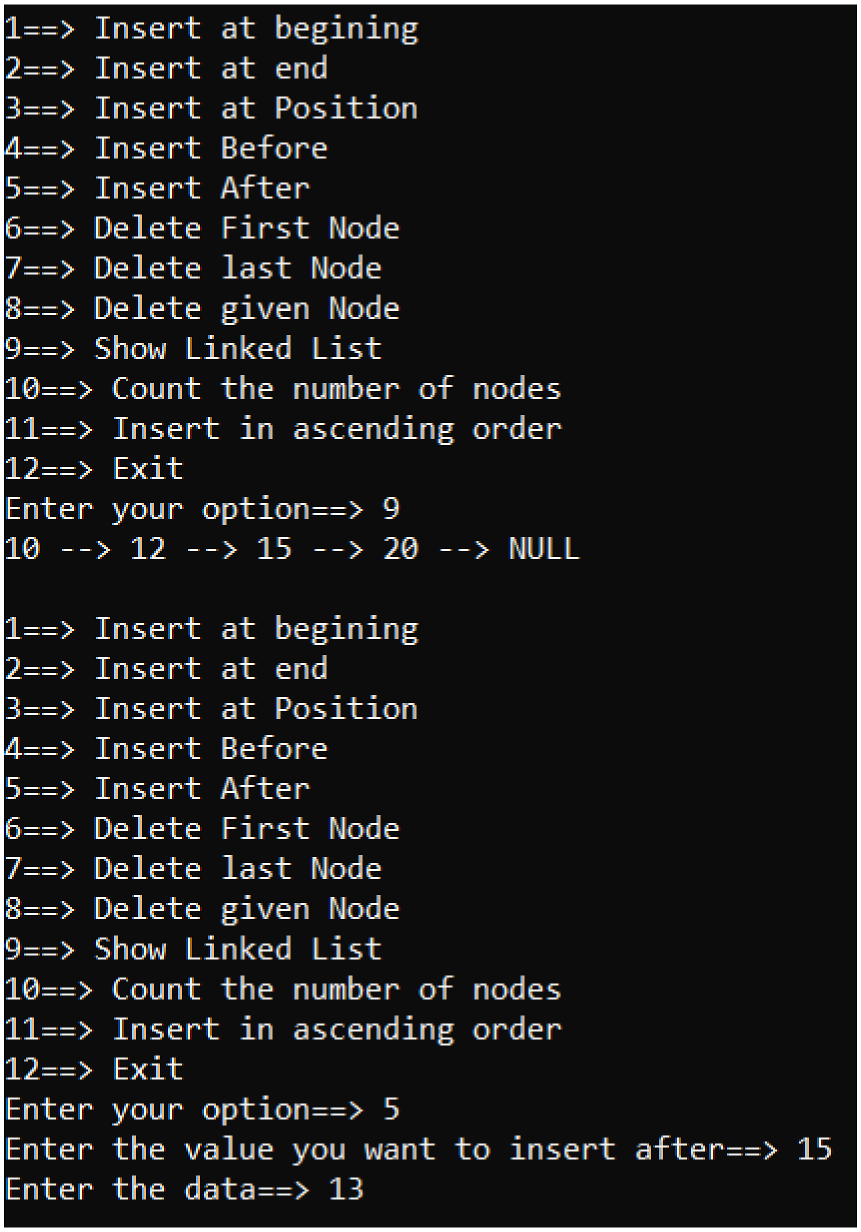
```
1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 9
10 --> 20 --> NULL

1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 3

Enter the index you want to insert your data at==> 1
Enter the data for Node==> 15
```

Roll No: 40 **Name: Akshit Trivedi**

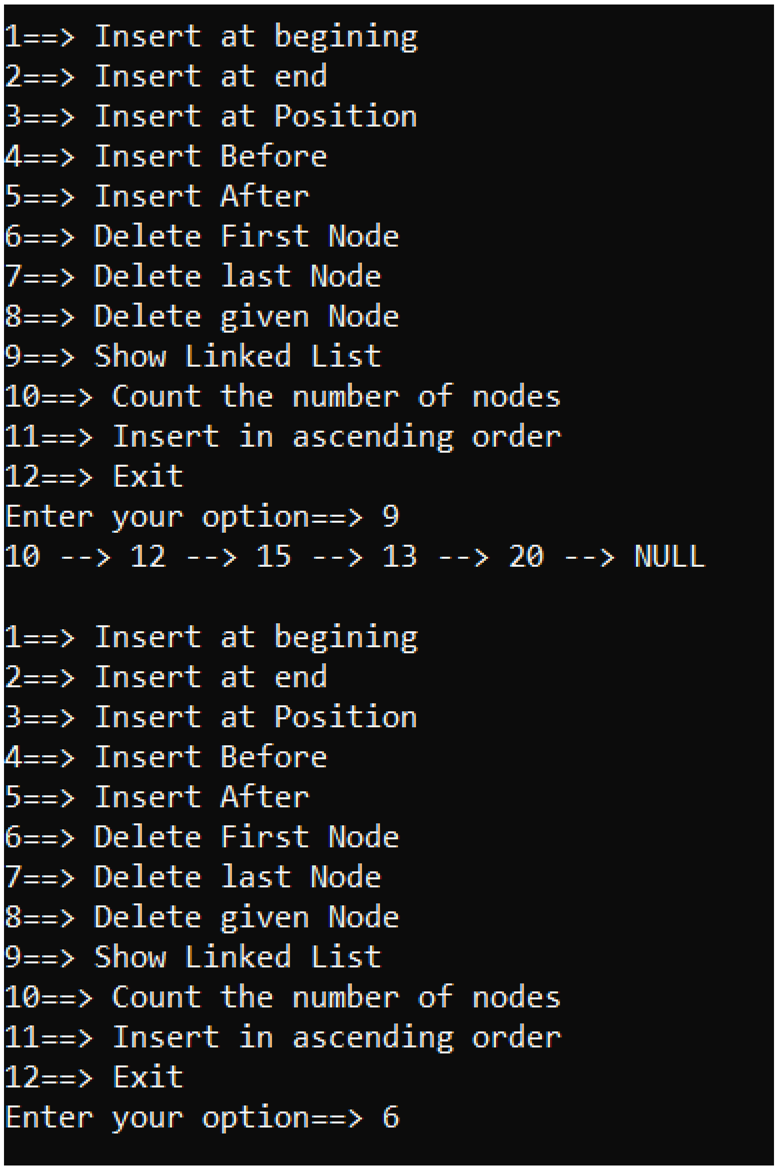
Class: MCA-1 **Year: 2021-22**



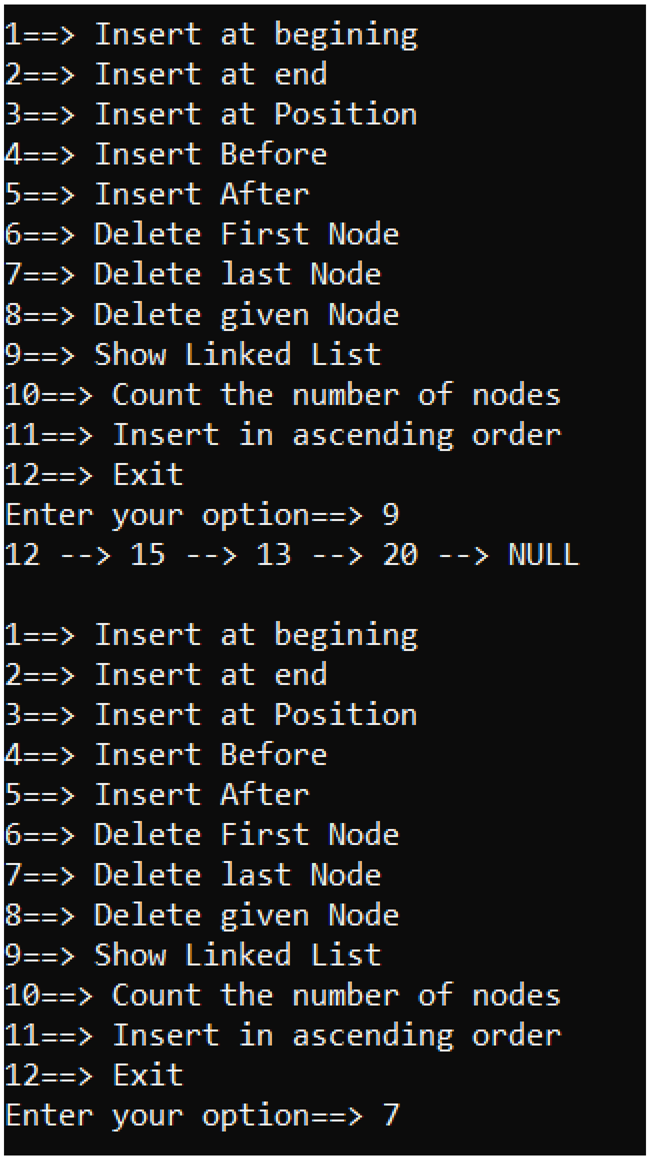
```
1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 9
10 --> 15 --> 20 --> NULL

1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 4
Enter the value you want to insert before==> 15
Enter the data==> 12
```



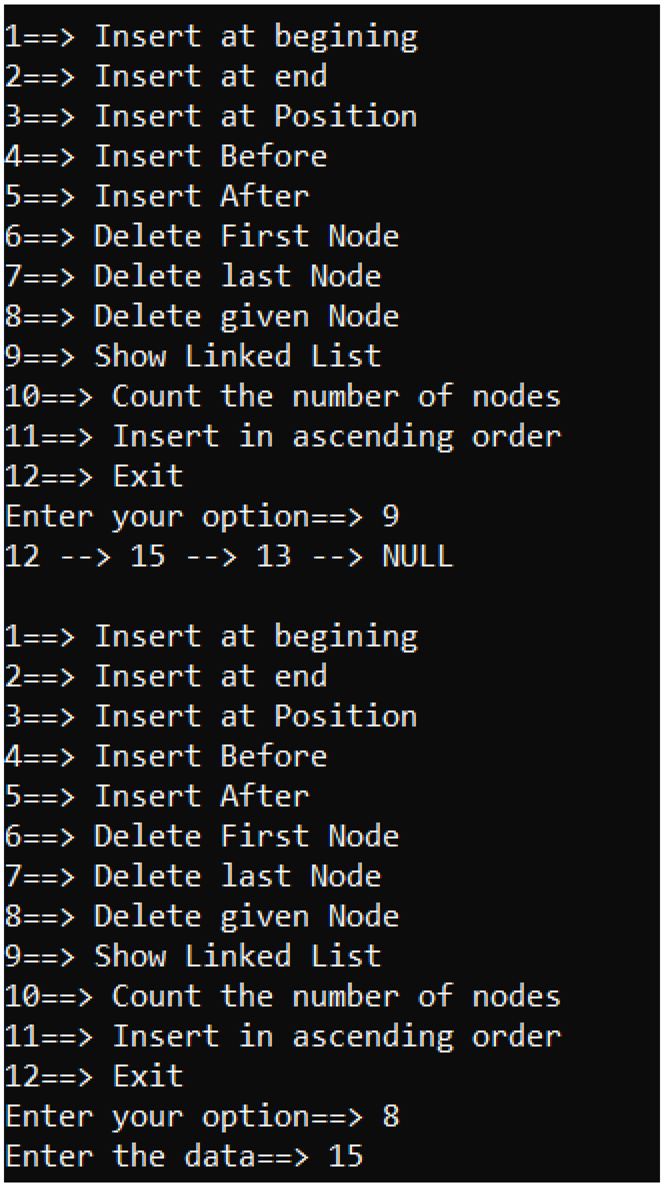
```
1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 9
10 --> 12 --> 15 --> 20 --> NULL

1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 5
Enter the value you want to insert after==> 15
Enter the data==> 13
```



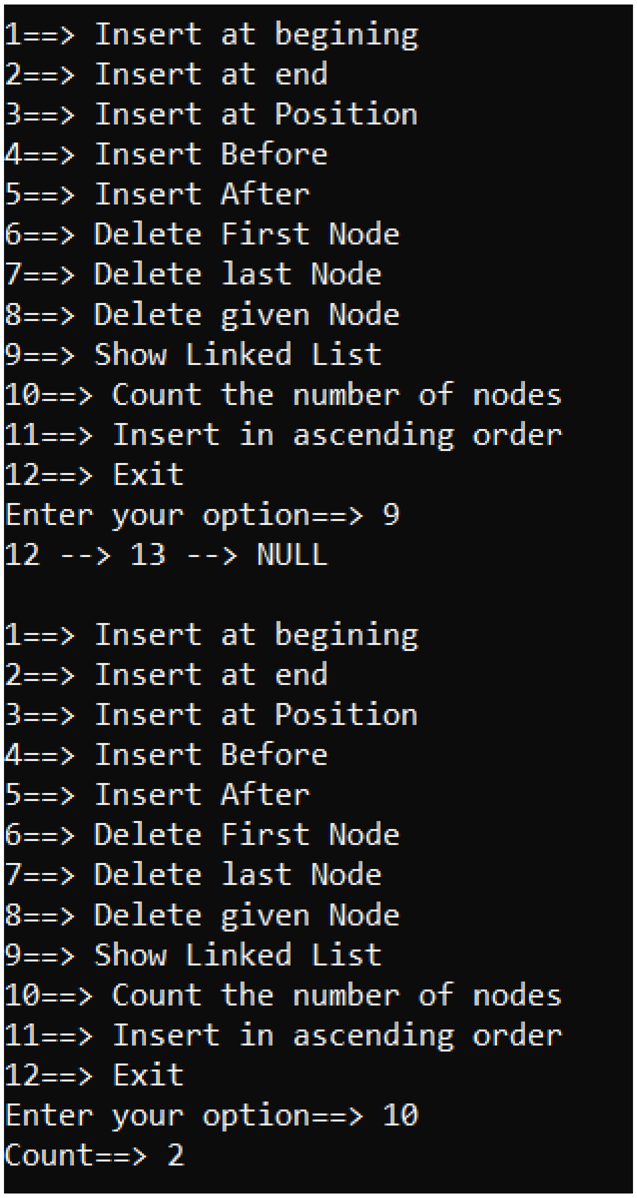
```
1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 9
10 --> 12 --> 15 --> 13 --> 20 --> NULL

1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 6
```



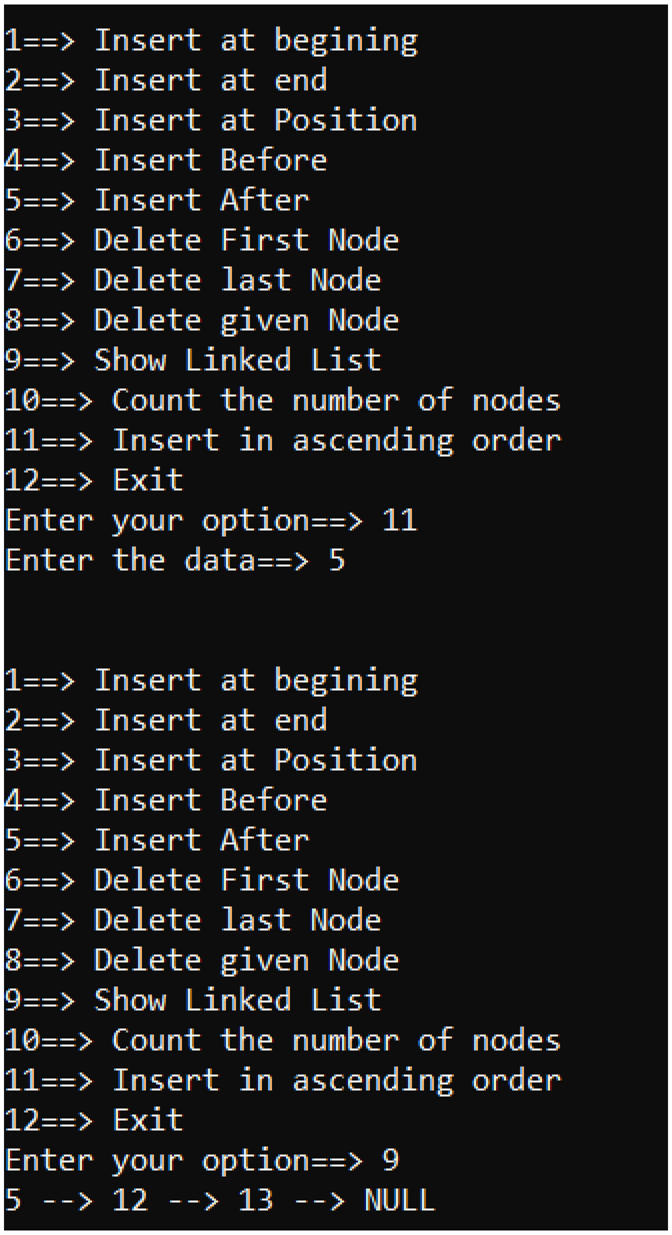
```
1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 9
12 --> 15 --> 13 --> 20 --> NULL

1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 7
```



```
1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 9
12 --> 15 --> 13 --> NULL

1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 8
Enter the data==> 15
```



```
1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 9
12 --> 13 --> NULL

1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 10
Count==> 2
```



```
1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 11
Enter the data==> 5


1==> Insert at begining
2==> Insert at end
3==> Insert at Position
4==> Insert Before
5==> Insert After
6==> Delete First Node
7==> Delete last Node
8==> Delete given Node
9==> Show Linked List
10==> Count the number of nodes
11==> Insert in ascending order
12==> Exit
Enter your option==> 9
5 --> 12 --> 13 --> NULL
```

DS ASSIGNMENT

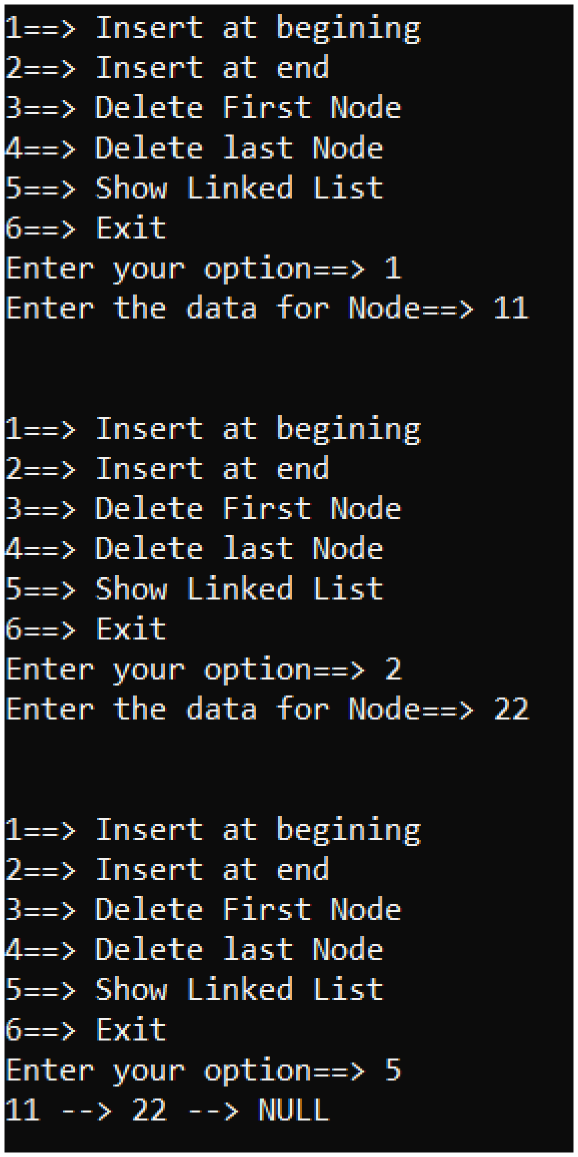
Roll No: 40 **Name: Akshit Trivedi**
Class: MCA-1 **Year: 2021-22**

2. Circular Linked List.

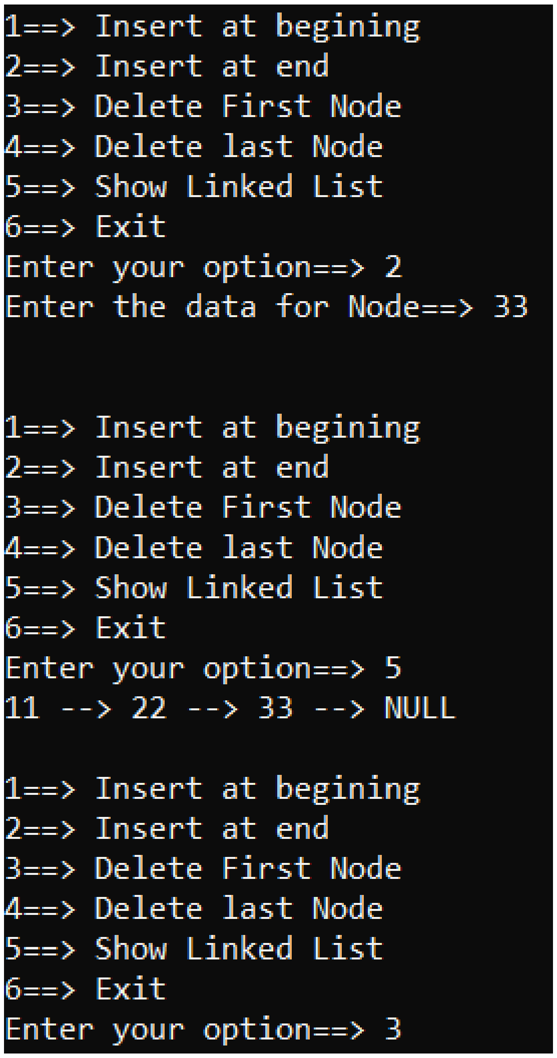
INPUT:

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>


struct node

{

    int data;

    struct node *next;

}*new_node, *temp, *head, *tail;


void create()
{


}


void insert_at_begin()

{

    int data;

    new_node = (struct node*)malloc(sizeof(struct node));

    printf("Enter the data for Node==> ");

    scanf("%d",&data);

    new_node->data = data;


    if(head==NULL)
    {
        new_node->next = new_node;
        head = new_node;
        tail = new_node;
    }
    else
    {
        new_node->next = head;
        head = new_node;
        tail->next = head;
    }
}

void insert_at_end()
{
    int data;
    new_node = (struct node*)malloc(sizeof(struct node));
    printf("Enter the data for Node==> ");
    scanf("%d",&data);
    new_node->data = data;

    if(head==NULL)
    {
        new_node->next = new_node;
        head = new_node;
        tail = new_node;
    }
    else
    {
        int data;
        temp = head;
        while(temp->next != head)
        {
            temp = temp->next;
        }
        new_node->next = head;
        temp->next = new_node;
        tail = new_node;
    }
}


void del_first()
{
    if(head==NULL)
    {
        printf("Linked list is empty");
        return;
    }
    if(head->next == head)
    {
        free(head);
        tail = NULL;
        head = NULL;
        return;
    }
    temp = head->next;
    free(head);
    head = temp;
    tail->next = head;
}

void del_last()
{
    struct node *preptr, *ptr;
    if(head == NULL)
    {
        printf("LInked list is empty");
        return;
    }
    if(head->next==head)
    {
        free(head);
        head = NULL;
        tail = NULL;
        return;
    }

    ptr = head;
    preptr = head;

    while(ptr->next != head)
    {
        preptr = ptr;
        ptr = ptr->next;
    }
    free(ptr);
    preptr->next = head;
    tail = preptr;
}

void show()
{
    if(head==NULL)
    {
        printf("Linked List is empty");
        return;
    }
    temp = head;

    do
    {
        printf("%d --> ", temp->data);
        temp = temp->next;
    }while(temp!=head);
    printf("NULL");
}

void insert_ascending()
{
    int data;
    struct node *ptr, *preptr;
    ptr = head;
    preptr = head;
    if(head == NULL)
    {
        create();
        return;
    }
    printf("Enter the data==> ");
    scanf("%d", &data);
    new_node = (struct node*)malloc(sizeof(struct node));
    while(ptr != NULL && ptr->data <= data )
    {
        preptr = ptr;
        ptr = ptr->next;
    }
    new_node->data = data;
    new_node->next = ptr;
    if(head->data >= data)
    {
        head = new_node;
        return;
    }
    preptr->next = new_node;
}

void main()
{
    int op=0;
    while(op!=12)
    {
        printf("\n\n1==> Insert at begining");
        printf("\n2==> Insert at end");
        printf("\n3==> Delete First Node");
        printf("\n4==> Delete last Node");
        printf("\n5==> Show Linked List");
        printf("\n6==> Exit");
        printf("\nEnter your option==> ");
        scanf("%d",&op);
        switch (op)
        {
            case 1 : insert_at_begin();
                break;

            case 2 : insert_at_end();
                break;
            case 3 : del_first();

                break;


            case 4 : del_last();

                break;


            case 5 : show();

                break;


            case 6 : printf("\nExiting!!!");

                break;


            default : printf("\nInvalid Input");

                break;

        }

    }

}
```

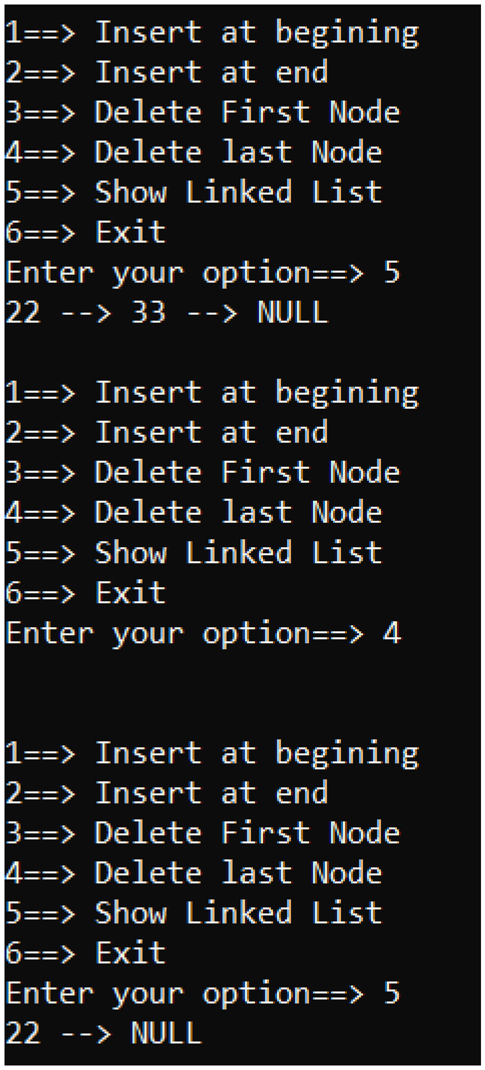
OUTPUT:



```
1==> Insert at begining
2==> Insert at end
3==> Delete First Node
4==> Delete last Node
5==> Show Linked List
6==> Exit
Enter your option==> 1
Enter the data for Node==> 11


1==> Insert at begining
2==> Insert at end
3==> Delete First Node
4==> Delete last Node
5==> Show Linked List
6==> Exit
Enter your option==> 2
Enter the data for Node==> 22


1==> Insert at begining
2==> Insert at end
3==> Delete First Node
4==> Delete last Node
5==> Show Linked List
6==> Exit
Enter your option==> 5
11 --> 22 --> NULL
```



```
1==> Insert at begining
2==> Insert at end
3==> Delete First Node
4==> Delete last Node
5==> Show Linked List
6==> Exit
Enter your option==> 2
Enter the data for Node==> 33


1==> Insert at begining
2==> Insert at end
3==> Delete First Node
4==> Delete last Node
5==> Show Linked List
6==> Exit
Enter your option==> 5
11 --> 22 --> 33 --> NULL

1==> Insert at begining
2==> Insert at end
3==> Delete First Node
4==> Delete last Node
5==> Show Linked List
6==> Exit
Enter your option==> 3
```


```
1==> Insert at begining
2==> Insert at end
3==> Delete First Node
4==> Delete last Node
5==> Show Linked List
6==> Exit
Enter your option==> 5
22 --> 33 --> NULL

1==> Insert at begining
2==> Insert at end
3==> Delete First Node
4==> Delete last Node
5==> Show Linked List
6==> Exit
Enter your option==> 4


1==> Insert at begining
2==> Insert at end
3==> Delete First Node
4==> Delete last Node
5==> Show Linked List
6==> Exit
Enter your option==> 5
22 --> NULL
```

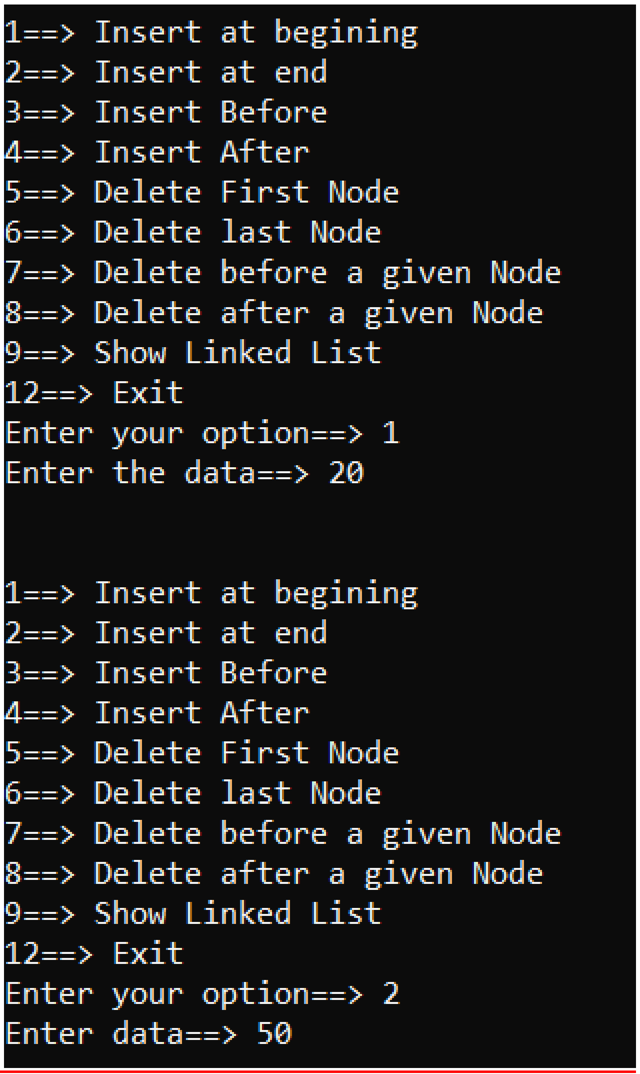
3. Doubly Linked List.

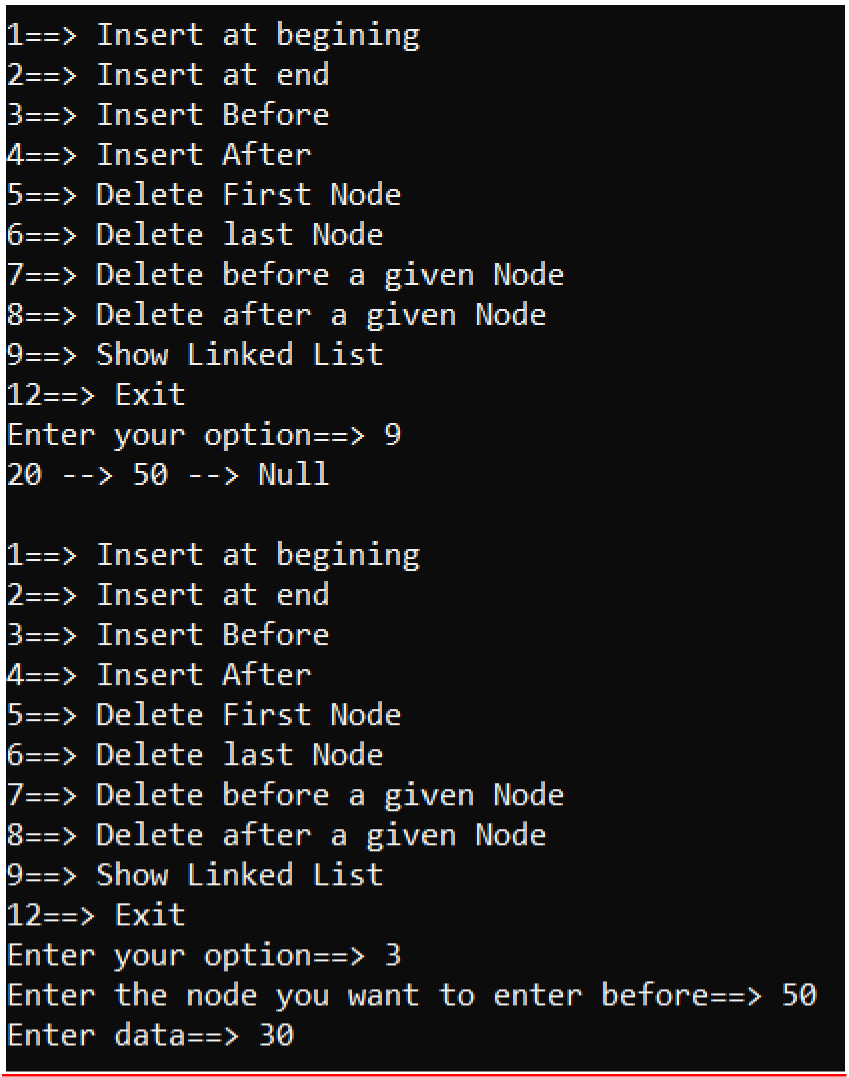
INPUT:

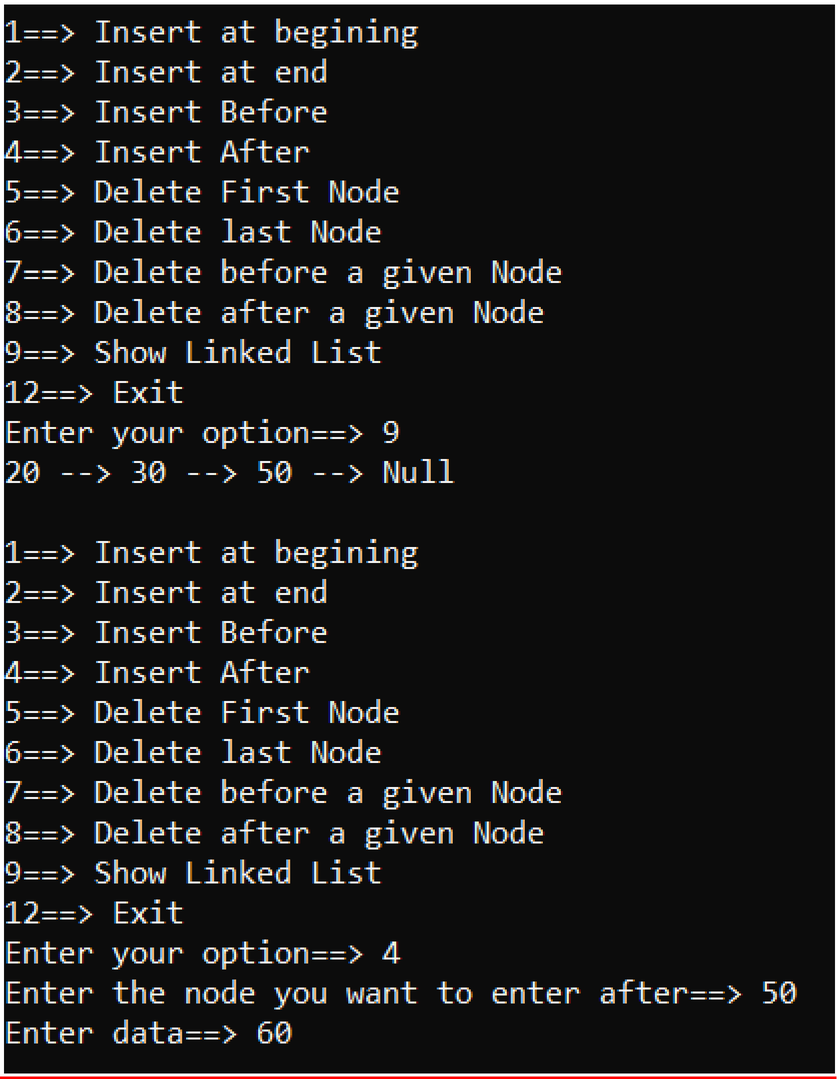
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

struct node
{
    int data;
    struct node *prev;
    struct node *next;
}*head, *new_node;

void create()
{
    int data;
    printf("Enter the data==> ");
    scanf("%d",&data);
    new_node = (struct node*)malloc(sizeof(struct node));
    new_node->data = data;
    new_node->prev = NULL;
    new_node->next =NULL;
    head = new_node;
}

void insert_at_beg()
{

    if (head==NULL)
    {
        create();
        return;
    }

    int data;
    struct node *ptr;
    ptr = head;
    printf("Enter the data==> ");
    scanf("%d", &data);
    new_node = (struct node*) malloc(sizeof(struct node));
    new_node->data = data;
    new_node->prev = NULL;
    new_node->next = head;
    head->prev= new_node;
    head = new_node;
}

void insert_at_end()
{
    if(head==NULL)
    {
        insert_at_beg();
        return;
    }

    struct node *ptr;
    ptr = head;
    int data;
    while(ptr->next != NULL)
    {
        ptr = ptr->next;
    }

    printf("Enter data==> ");
    scanf("%d", &data);
    new_node = (struct node*)malloc(sizeof(struct node));
    new_node->data = data;
    new_node->prev = ptr;
    new_node->next = NULL;
    ptr->next = new_node;
}

void insert_before()
{
    if(head==NULL)
    {
        printf("Linked List is empty");
        return;
    }
    int before, data, flag=0;
    struct node *ptr;
    struct node * preptr;
    ptr = head;
    printf("Enter the node you want to enter before==> ");
    scanf("%d", &before);
    if(head->data == before)
    {
        insert_at_beg();
        return;
    }

    while(ptr != NULL)
    {
        if(ptr->data == before)
        {
            flag=1;
            break;
        }
        preptr = ptr;
        ptr = ptr->next;
    }

    if(flag==1)
    {
        printf("Enter data==> ");
        scanf("%d", &data);
        new_node = (struct node*)malloc(sizeof(struct node));
        new_node->data = data;
        new_node->next = ptr;
        new_node->prev = preptr;
        preptr->next = new_node;
        ptr->prev = new_node;
    }
    else
    {
        printf("%d does not exist in the linked list", before);
    }

}

void insert_after()
{
    if(head==NULL)
    {
        printf("Linked List is empty");
        return;
    }
    int after, data, flag=0;
    struct node *ptr;
    struct node *preptr;
    ptr = head;
    preptr = head;
    printf("Enter the node you want to enter after==> ");
    scanf("%d", &after);
    if(head->data == after)
    {
        insert_at_end();
        return;
    }

    while(ptr != NULL)
    {
        if(ptr->data == after)
        {
            flag=1;
            break;
        }
        ptr = ptr->next;
    }

    if(flag)
    {
        printf("Enter data==> ");
        scanf("%d", &data);
        new_node = (struct node*)malloc(sizeof(struct node));
        new_node->data = data;
        new_node->next = ptr->next;
        new_node->prev = ptr;
        ptr->next = new_node;
    }
    else
    {
        printf("%d does not exist in the linked list", after);
    }

}

void delete_first()
{
    if(head==NULL)
    {
        printf("Linked list is empty");
        return;
    }
    struct node *ptr;
    ptr = head->next;
    ptr->prev = NULL;
    free(head);
    head = ptr;
}

void delete_last()
{
    if(head==NULL)
    {
        printf("Linked List is empty");
        return;
    }

    if(head->next == NULL)
    {
        free(head);
        return;
    }
    struct node *ptr;
    struct node *preptr;
    ptr = head;

    while(ptr->next != NULL)
    {
        preptr = ptr;
        ptr = ptr->next;
    }
    preptr->next = NULL;
    free(ptr);
}

void delete_before()
{
    struct node *ptr;
    struct node *preptr;
    int val_search, flag;

    if(head==NULL)
    {
        printf("Empty LInked List");
    }
    printf("Enter the data you want to delete before==> ");
    scanf("%d", &val_search);

    ptr = head;
    preptr = head;

    if(ptr->next->data == val_search)
    {
        delete_first();
        return;
    }

    while(ptr != NULL)
    {
        if(ptr->data == val_search)
        {
            flag = 1;
            break;
        }
        preptr = ptr;
        ptr = ptr->next;
    }
    preptr->next = ptr->next;
    if(ptr->next != NULL)
    {
        ptr->next->prev = preptr;
    }
    free(ptr);
}

void delete_after()
{
    struct node *ptr;
    struct node *preptr;
    int val_search, flag;
    if(head==NULL)
    {
        printf("Empty LInked List");
        return;
    }

    printf("Enter the data you want to delete after==> ");
    scanf("%d", &val_search);

    if(head->data == val_search)
    {
        delete_first();
        return;
    }

    ptr = head;
    preptr = head;

    while(ptr != NULL)
    {
        if(preptr->data == val_search)
        {
            flag = 1;
            break;
        }
        preptr = ptr;
        ptr = ptr->next;
    }

    if(!flag)
    {
        printf("%d not found in the linked list", val_search);
        return;
    }

    if(preptr->next == NULL)
    {
        printf("%d has no elements after to be deleted", val_search);
        return;
    }

    preptr->next = ptr->next;
    if(ptr->next != NULL)
    {
        ptr->next->prev = preptr;
    }
    free(ptr);
}

void show()
{
    struct node *ptr;
    ptr = head;
    while(ptr != NULL)
    {
        printf("%d",ptr->data);
        printf(" --> ");
        ptr = ptr->next;
    }
    printf("Null");
}

void main()
{
    int op = 0;
    while(op != 12)
    {
        printf("\n\n1==> Insert at begining");
        printf("\n2==> Insert at end");
        printf("\n3==> Insert Before");
        printf("\n4==> Insert After");
        printf("\n5==> Delete First Node");
        printf("\n6==> Delete last Node");
        printf("\n7==> Delete before a given Node");
        printf("\n8==> Delete after a given Node");
        printf("\n9==> Show Linked List");
        printf("\n12==> Exit");
        printf("\nEnter your option==> ");
        scanf("%d",&op);

        switch(op)
        {
            case 1:
                    insert_at_beg();
                    break;

            case 2:
                    insert_at_end();
                    break;

            case 3:
                    insert_before();
                    break;

            case 4:
                    insert_after();
                    break;
```

```
                case 5:
                        delete_first();
                        break;

                case 6:
                        delete_last();
                        break;

                case 7:
                        delete_before();
                        break;

                case 8:
                        delete_after();
                        break;

                case 9:
                        show();
                        break;

                case 12:
                        printf("Exiting!!!");
                        break;

                default:
                        printf("Invalid choice");

            }
        }
    }
```

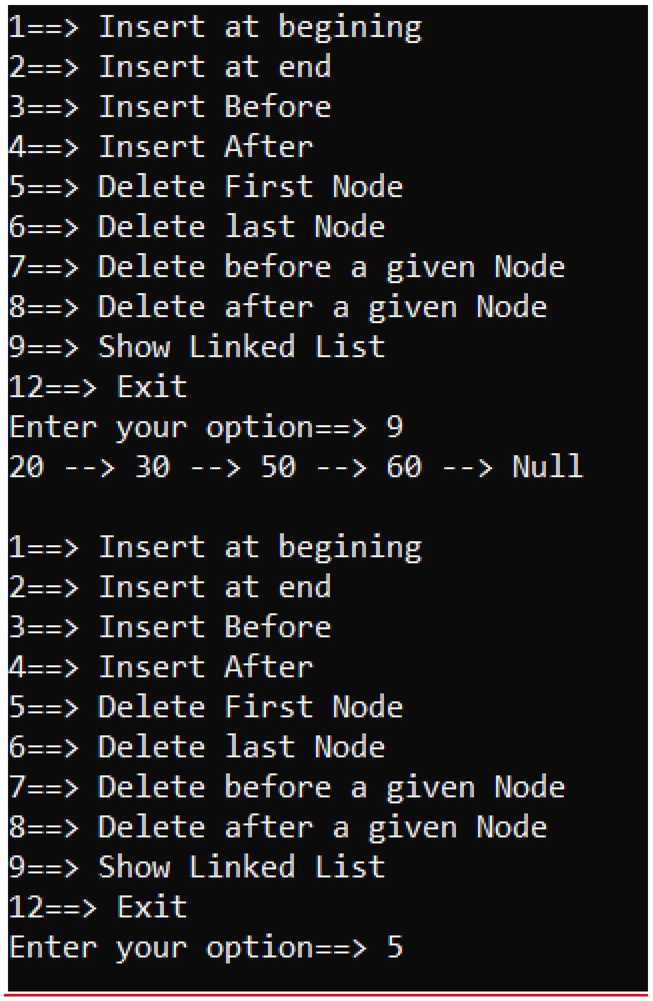
OUTPUT:


```
1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 1
Enter the data==> 20


1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 2
Enter data==> 50
```

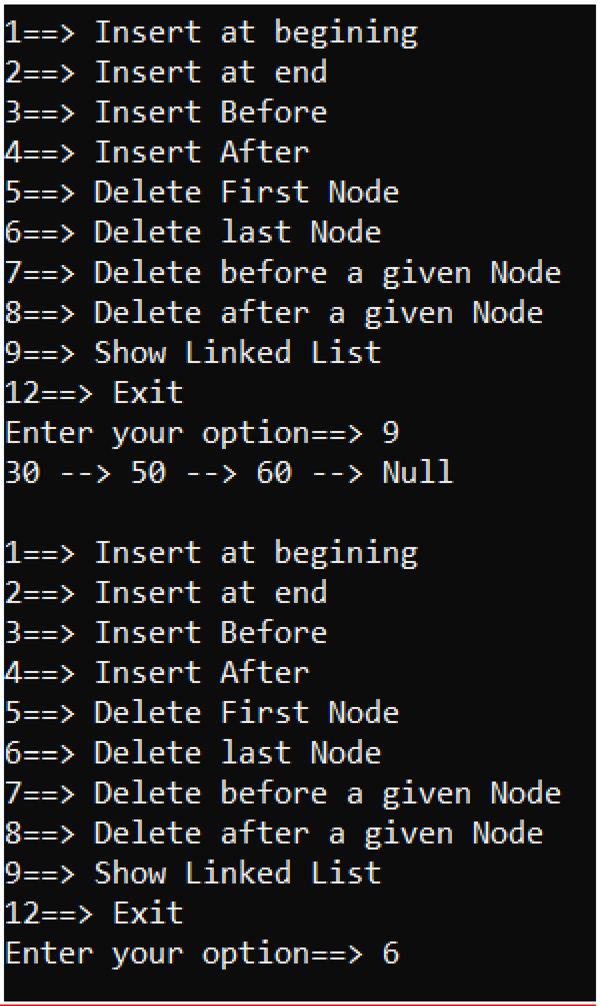

```
1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 9
20 --> 50 --> Null

1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 3
Enter the node you want to enter before==> 50
Enter data==> 30
```



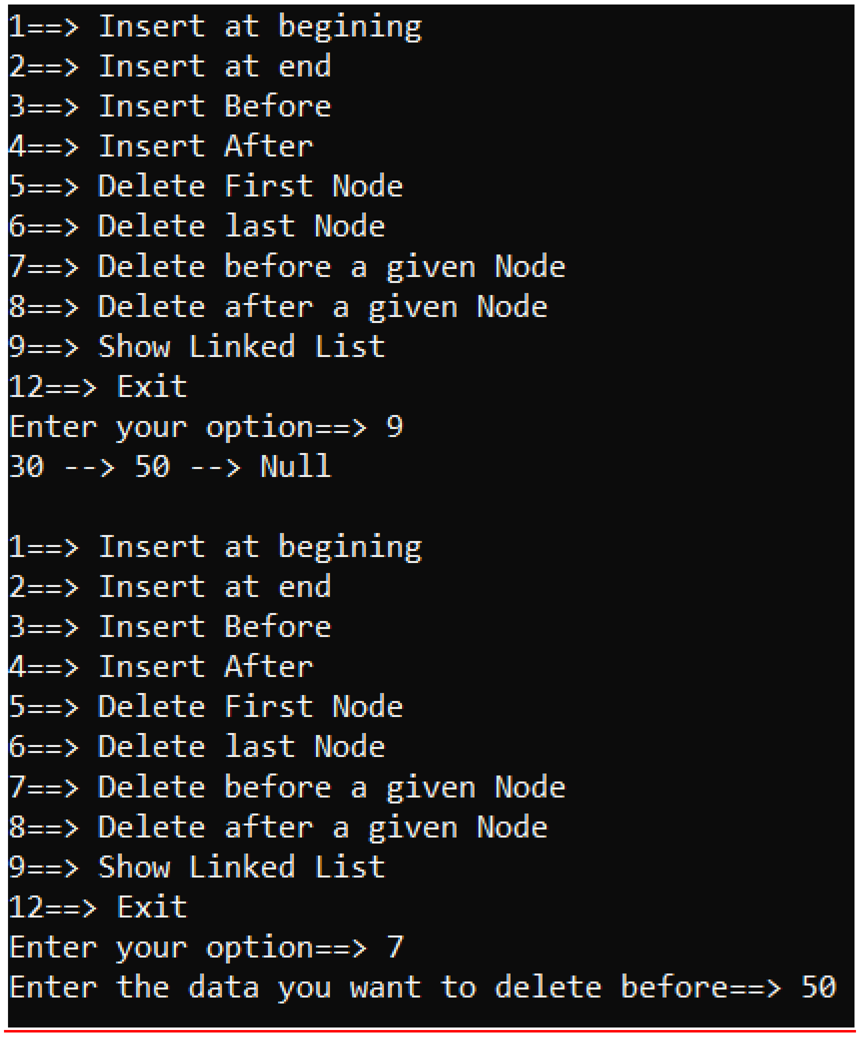
```
1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 9
20 --> 30 --> 50 --> Null

1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 4
Enter the node you want to enter after==> 50
Enter data==> 60
```



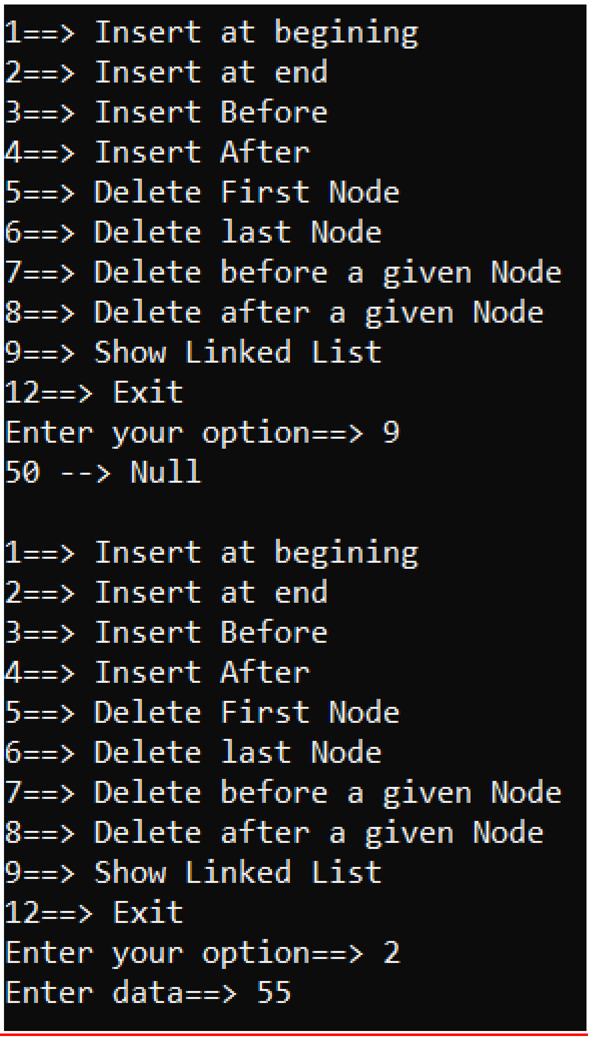
```
1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 9
20 --> 30 --> 50 --> 60 --> Null

1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 5
```



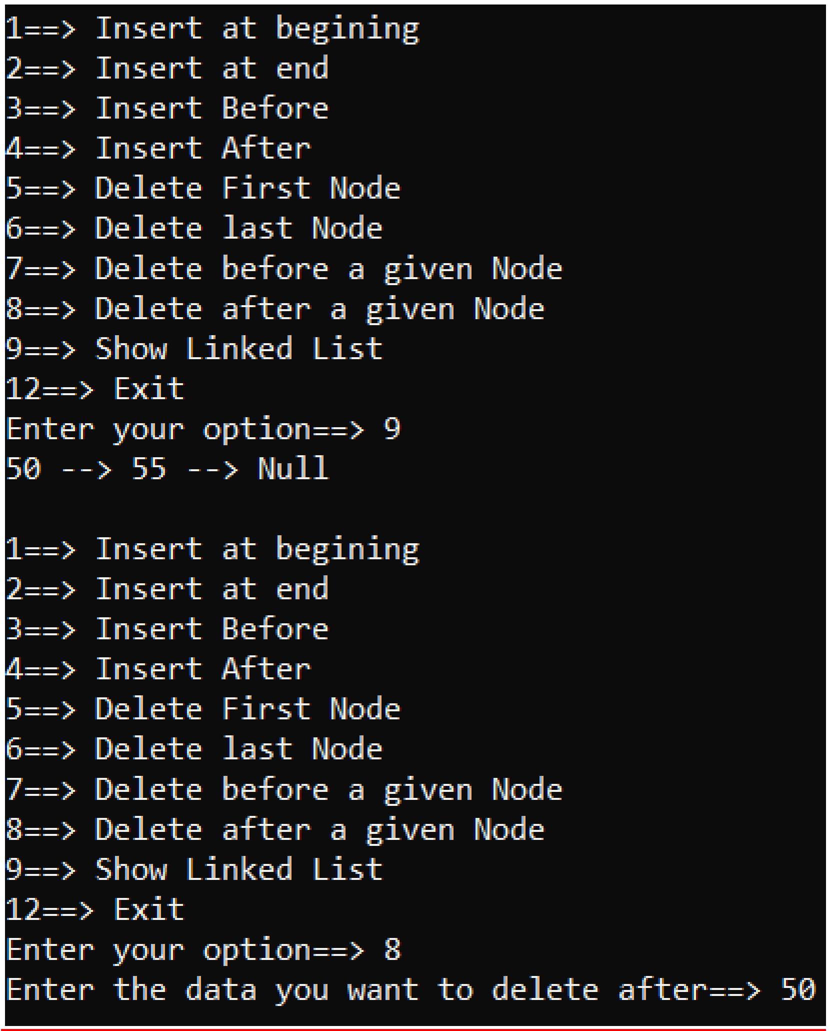
```
1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 9
30 --> 50 --> 60 --> Null

1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 6
```



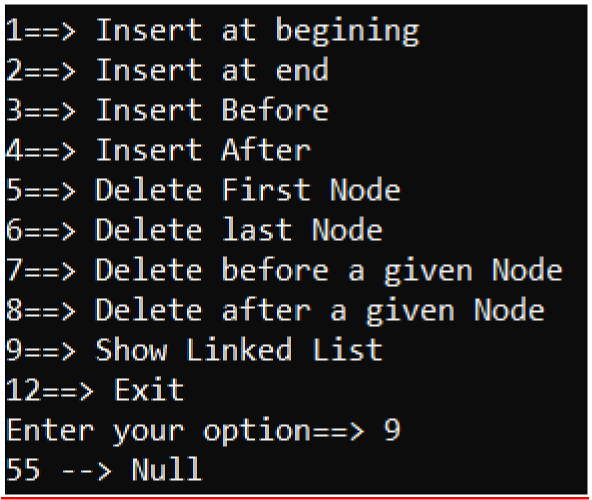
```
1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 9
30 --> 50 --> Null

1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 7
Enter the data you want to delete before==> 50
```



```
1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 9
50 --> Null

1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 2
Enter data==> 55
```



```
1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 9
50 --> 55 --> Null

1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 8
Enter the data you want to delete after==> 50
```



```
1==> Insert at begining
2==> Insert at end
3==> Insert Before
4==> Insert After
5==> Delete First Node
6==> Delete last Node
7==> Delete before a given Node
8==> Delete after a given Node
9==> Show Linked List
12==> Exit
Enter your option==> 9
55 --> Null
```

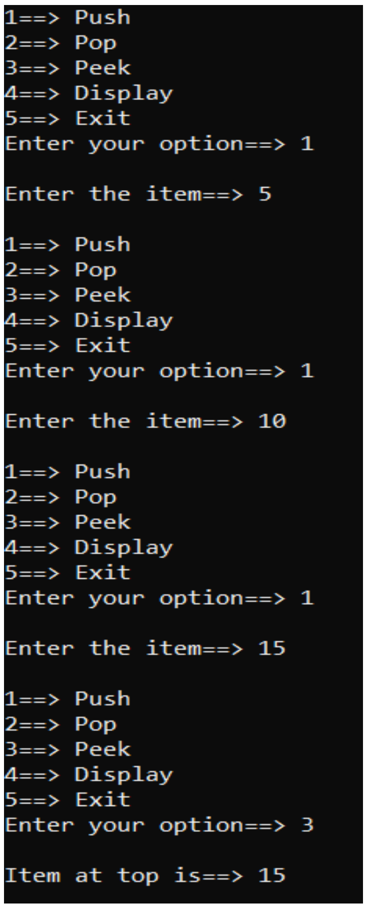
4. Stack using Array.

INPUT:

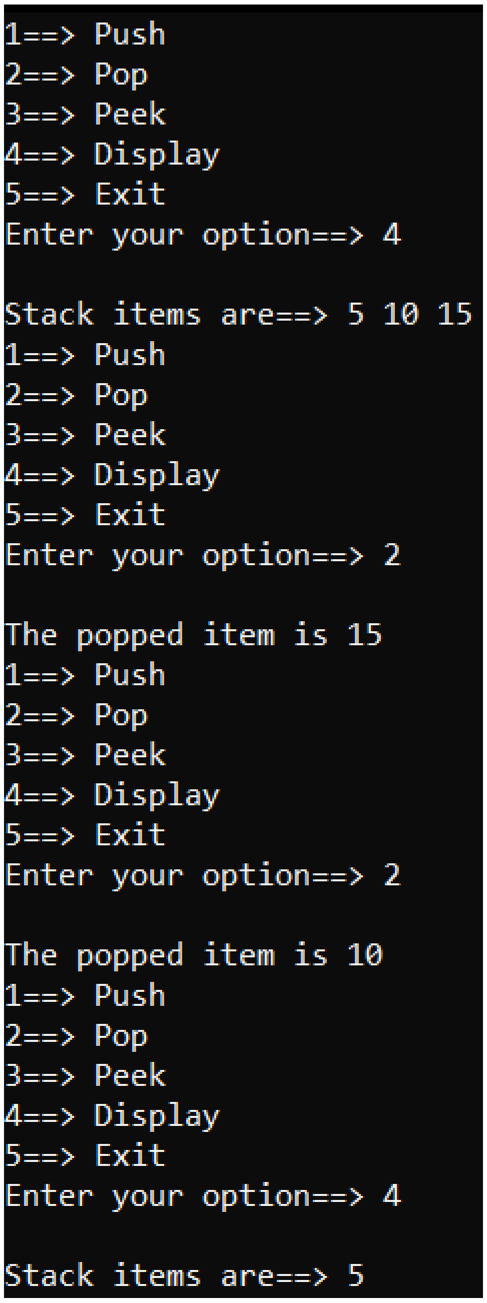
```
#include<stdio.h>
#include<conio.h>

int stack[5];
int max = 5;
int top=-1;

void push()
{
    int item;
    if(top>=max-1)
    {
        printf("\nStack overflow");
        return;
    }
    printf("\nEnter the item==> ");
    scanf("%d",&item);
    stack[++top] = item;
}

void pop()
{
    if(top<0)
    {
        printf("\nStack underflow");
        return;
    }
    printf("\nThe popped item is %d", stack[top--]);
}

void peek()
{
    if(top<0)
    {
        printf("\nStack underflow");
        return;
    }
    printf("\nItem at top is==> %d", stack[top]);
}

void display()
{
    int i;
    if(top<0)
    {
        printf("\nStack underflow");
        return;
    }
    printf("\nStack items are==> ");
    for(i=0;i<=top;i++)
    {
        printf("%d ", stack[i]);
    }
}

void main()
{
    int op=0;
    while(op!=5)
    {
        printf("\n1==> Push");
        printf("\n2==> Pop");
        printf("\n3==> Peek");
        printf("\n4==> Display");
        printf("\n5==> Exit");
        printf("\nEnter your option==> ");
        scanf("%d",&op);

        switch(op)
        {
            case 1: push();
                    break;

            case 2: pop();
```

```
                    break;

            case 3: peek();
                    break;

            case 4: display();
                    break;

            case 5: printf("Exiting the program!!!");
                    break;

            default: printf("Enter valid option");
                    break;
        }
    }
}
```

OUTPUT:



```
1==> Push
2==> Pop
3==> Peek
4==> Display
5==> Exit
Enter your option==> 1

Enter the item==> 5

1==> Push
2==> Pop
3==> Peek
4==> Display
5==> Exit
Enter your option==> 1

Enter the item==> 10

1==> Push
2==> Pop
3==> Peek
4==> Display
5==> Exit
Enter your option==> 1

Enter the item==> 15

1==> Push
2==> Pop
3==> Peek
4==> Display
5==> Exit
Enter your option==> 3

Item at top is==> 15
```



```
1==> Push
2==> Pop
3==> Peek
4==> Display
5==> Exit
Enter your option==> 4

Stack items are==> 5 10 15
1==> Push
2==> Pop
3==> Peek
4==> Display
5==> Exit
Enter your option==> 2

The popped item is 15
1==> Push
2==> Pop
3==> Peek
4==> Display
5==> Exit
Enter your option==> 2

The popped item is 10
1==> Push
2==> Pop
3==> Peek
4==> Display
5==> Exit
Enter your option==> 4

Stack items are==> 5
```

DS ASSIGNMENT

Roll No: 40 **Name: Akshit Trivedi**

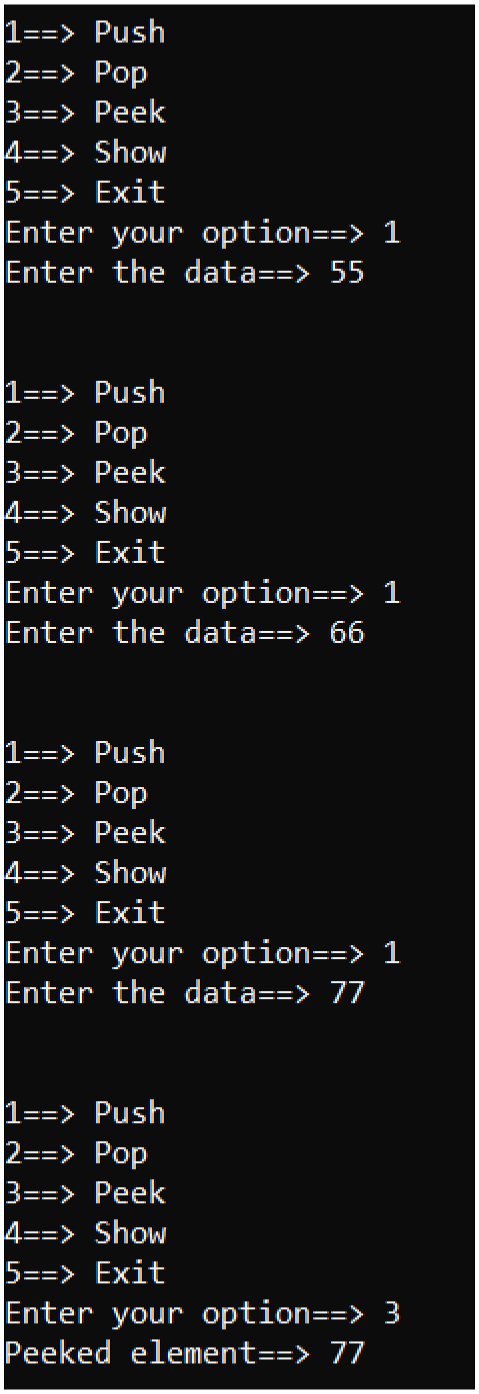
Class: MCA-1 **Year: 2021-22**

5. Stack Using Linked List.

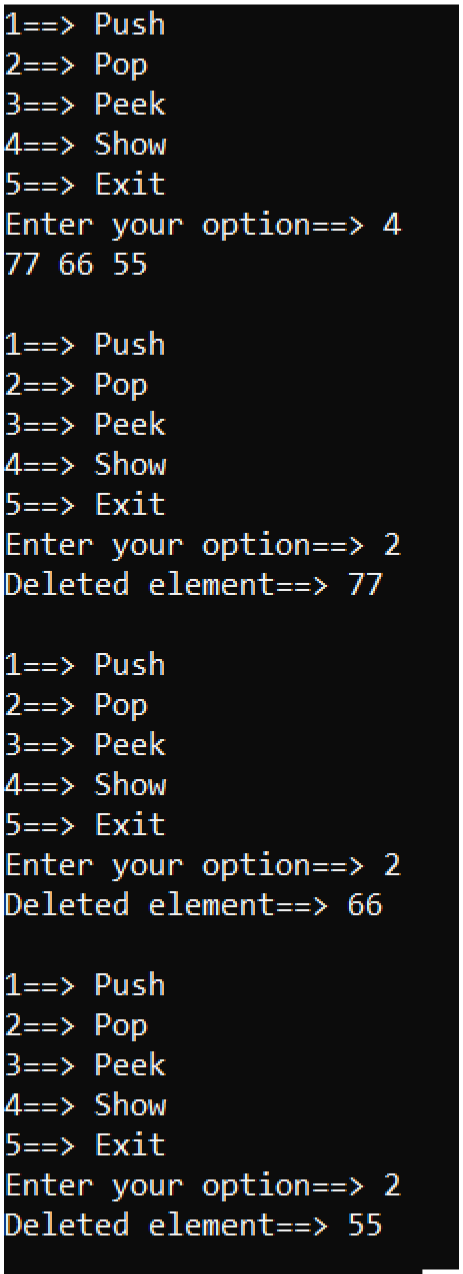
INPUT:

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

struct node
{
    int data;
    struct node *next;
}*new_node, *temp, *head;

void push()
{
    int data;
    new_node = (struct node*)malloc(sizeof(struct node));
    printf("Enter the data==> ");
    scanf("%d",&data);
    new_node->data = data;

    if(head==NULL)
    {
        new_node->next=NULL;
        head = new_node;
    }
    else
    {
        new_node->data = data;
        new_node->next = head;
        head = new_node;
    }
}

void pop()
{
    int data;
    if(head==NULL)
    {
        printf("Stack is underflow");
        return;
    }
    printf("Deleted element==> %d", head->data);
    temp = head->next;
    data = head->data;
    free(head);
    head = temp;
}

void peek()
{
    if(head==NULL)
    {
        printf("Stack is underflow");
        return;
    }
    printf("Peeked element==> %d", head->data);
}

void show()
{
    if(head==NULL)
    {
        printf("Stack underflow");
        return;
    }
    temp = head;
    while(temp!=NULL)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

void main()
{
    int op=0;
    while(op!=5)
    {
        printf("\n\n1==> Push");
        printf("\n2==> Pop");
        printf("\n3==> Peek");
        printf("\n4==> Show");
        printf("\n5==> Exit");
        printf("\nEnter your option==> ");
        scanf("%d",&op);
        switch (op)
        {
            case 1 : push();
                     break;

            case 2 : pop();
                     break;

            case 3 : peek();
                     break;

            case 4 : show();
                     break;

            case 5 : printf("\nExiting!!!");
                    break;

            default : printf("\nInvalid Input");
                break;
        }
    }
}
```

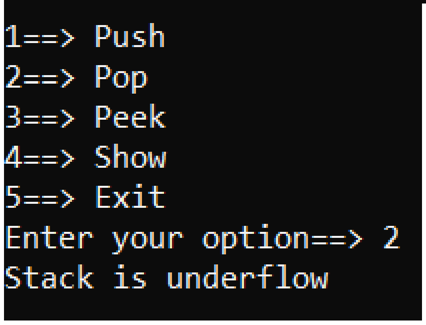
OUTPUT:



```
1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 1
Enter the data==> 55


1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 1
Enter the data==> 66


1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 1
Enter the data==> 77


1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 3
Peeked element==> 77
```



```
1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 4
77 66 55

1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 2
Deleted element==> 77

1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 2
Deleted element==> 66

1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 2
Deleted element==> 55
```



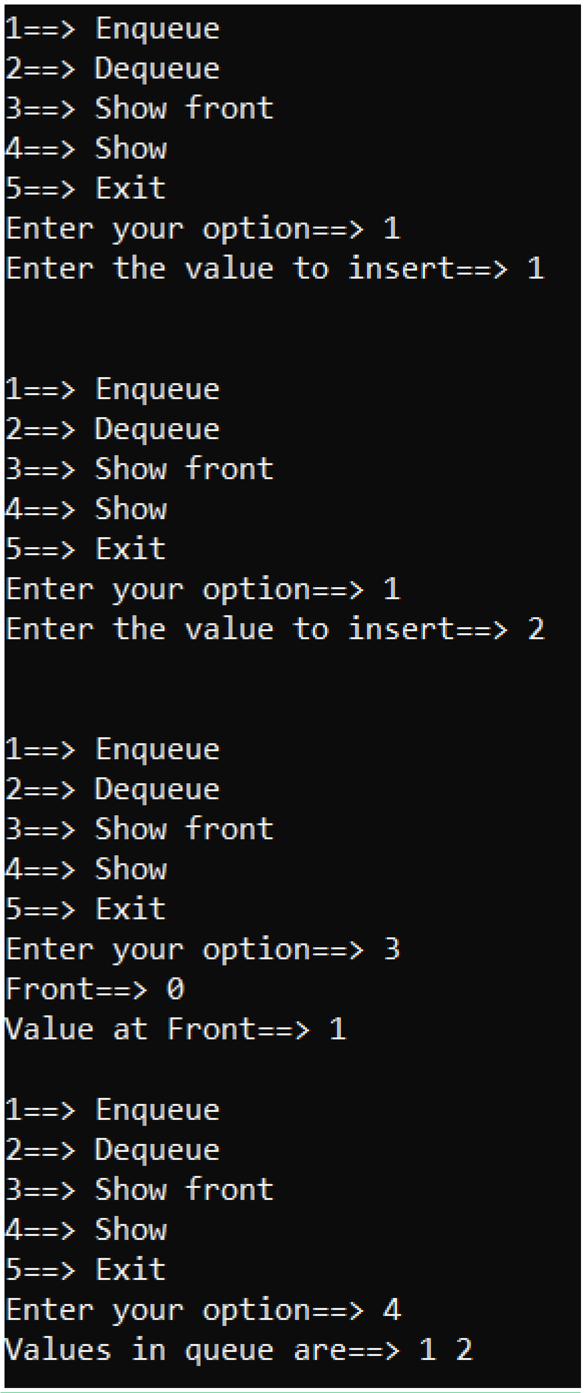
```
1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 2
Stack is underflow
```

6. Queue Using Array.

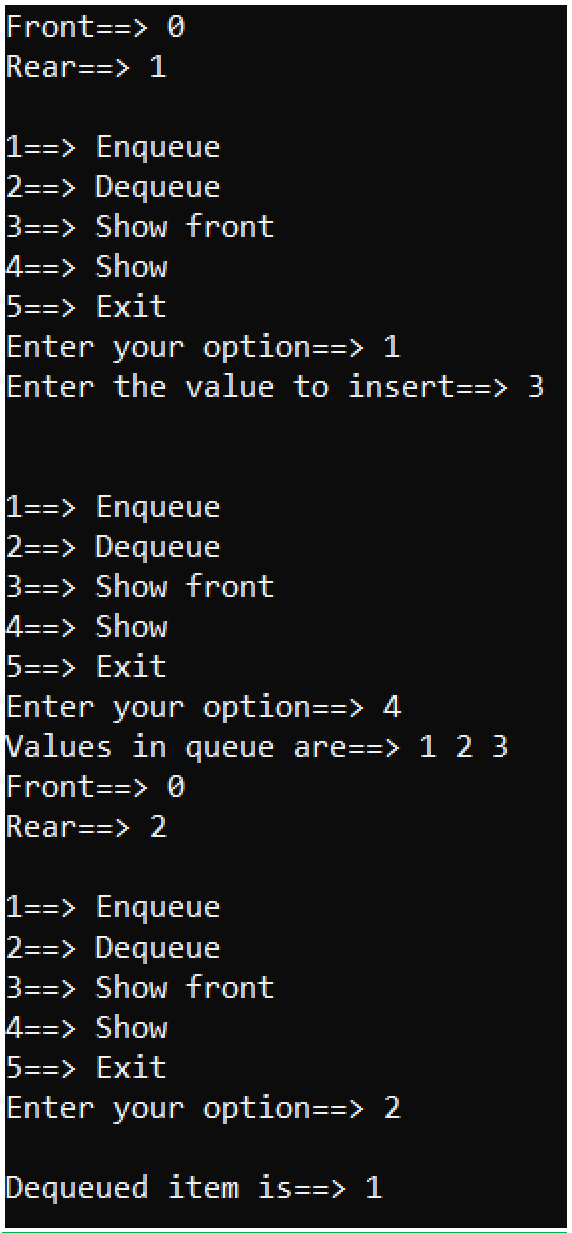
INPUT:

```
#include<stdio.h>
#include<conio.h>
int size=5;
int queue[5];
int front = -1;
int rear = -1;

void enqueue()
{
    int val;
    if(rear == size-1)
    {
        printf("Queue is full");
        return;
    }

    if(rear == -1 & front == -1 )
    {
        front += 1;
    }

    printf("Enter the value to insert==> ");
    scanf("%d", &val);
    rear +=1;
    queue[rear] = val;
}

void dequeue()
{
    if(rear==-1)
    {
        printf("Queue is empty");
        return;
    }
    printf("\nDequeued item is==> %d", queue[front++]);

    if(front>rear)
    {
        front=-1;
        rear=-1;
    }

}

void show()
{
    int i = 0;
    if(rear==-1)
    {
        printf("Queue is empty");
        return;
    }
    printf("Values in queue are==> ");
    for(i=front;i<=rear;i++)
    {
        printf("%d ", queue[i]);
    }
    printf("\nFront==> %d", front);
    printf("\nRear==> %d", rear);
}

void show_front()
{
    printf("Front==> %d", front);
    if(front!=-1) printf("\nValue at Front==> %d", queue[front]);
}

void main()
{
    int op = 0;
    while(op != 5)
    {
        printf("\n\n1==> Enqueue");
        printf("\n2==> Dequeue");
        printf("\n3==> Show front");
        printf("\n4==> Show");
        printf("\n5==> Exit");
        printf("\nEnter your option==> ");
        scanf("%d",&op);

        switch(op)
        {
            case 1:
                    enqueue();
                    break;

            case 2:
                    dequeue();
                    break;

            case 3:
                    show_front();
                    break;

            case 4:
                    show();
                    break;

            case 5:
                    printf("Exiting the program");
                    break;

            default: printf("Enter valid option");
                    break;
        }
    }

}
```

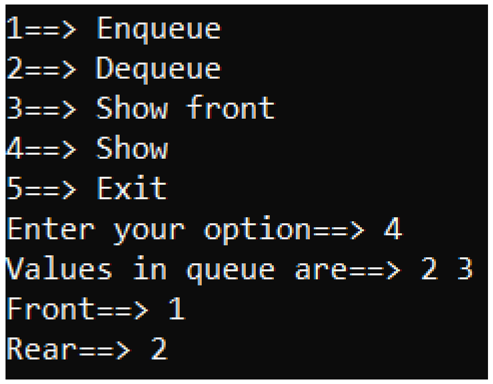
OUTPUT:



```
1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 1
Enter the value to insert==> 1


1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 1
Enter the value to insert==> 2


1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 3
Front==> 0
Value at Front==> 1

1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 4
Values in queue are==> 1 2
```



```
Front==> 0
Rear==> 1

1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 1
Enter the value to insert==> 3


1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 4
Values in queue are==> 1 2 3
Front==> 0
Rear==> 2

1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 2

Dequeued item is==> 1
```



```
1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 4
Values in queue are==> 2 3
Front==> 1
Rear==> 2
```

DS ASSIGNMENT

Roll No: 40 **Name: Akshit Trivedi**
Class: MCA-1 **Year: 2021-22**

7. Queue using Linked List.

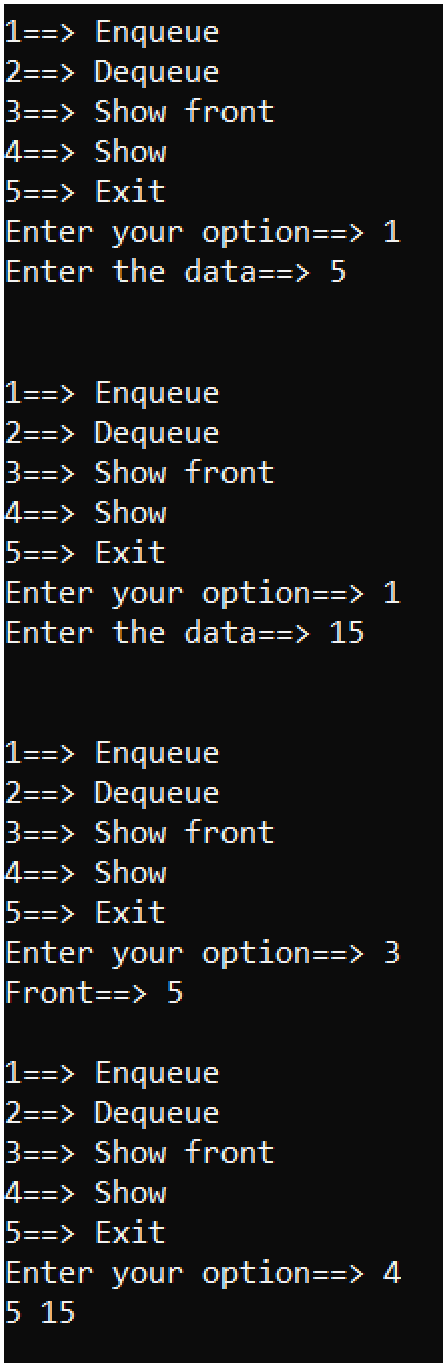
INPUT:

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

struct node
{
    int data;
    struct node *next;
}*new_node, *temp, *head, *tail;

void enqueue()
{
    int data;
    new_node = (struct node*)malloc(sizeof(struct node));
    printf("Enter the data==> ");
    scanf("%d",&data);

    if(head==NULL)
    {
        new_node->data = data;
        new_node->next=NULL;
        head = new_node;
    }
    else
    {
        temp = head;
        while(temp->next != NULL)  //Traversing to the end node
        {
            temp = temp->next;
        }
        new_node->data = data;
        new_node->next = NULL;
        temp->next = new_node;
    }
}
```

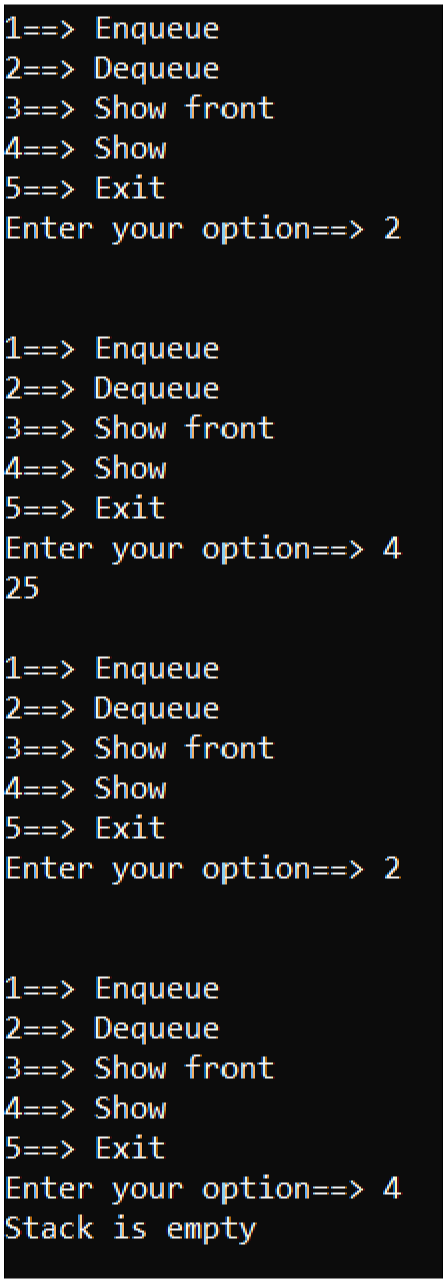
```c
void dequeue()
{
    if(head==NULL)
    {
        printf("Linked list is empty");
        return;
    }
    temp = head->next;
    free(head);
    head = temp;
}

void show()
{
    if(head==NULL)
    {
        printf("Stack is empty");
        return;
    }
    temp = head;
    while(temp!=NULL)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

void show_front()
{
    if(head==NULL)
    {
        printf("Stack is empty");
        return;
    }
    printf("Front==> %d", head->data);
}
```

```c
void main()
{
    int op = 0;
    while(op != 5)
    {
        printf("\n\n1==> Enqueue");
        printf("\n2==> Dequeue");
        printf("\n3==> Show front");
        printf("\n4==> Show");
        printf("\n5==> Exit");
        printf("\nEnter your option==> ");
        scanf("%d",&op);

        switch(op)
        {
            case 1:
                    enqueue();
                    break;

            case 2:
                    dequeue();
                    break;

            case 3:
                    show_front();
                    break;

            case 4:
                    show();
                    break;

            case 5:
                    printf("Exiting the program");
                    break;

            default: printf("Enter valid option");
                    break;
        }
    }
}
```

OUTPUT:



```
1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 1
Enter the data==> 5


1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 1
Enter the data==> 15


1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 3
Front==> 5

1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 4
5 15
```



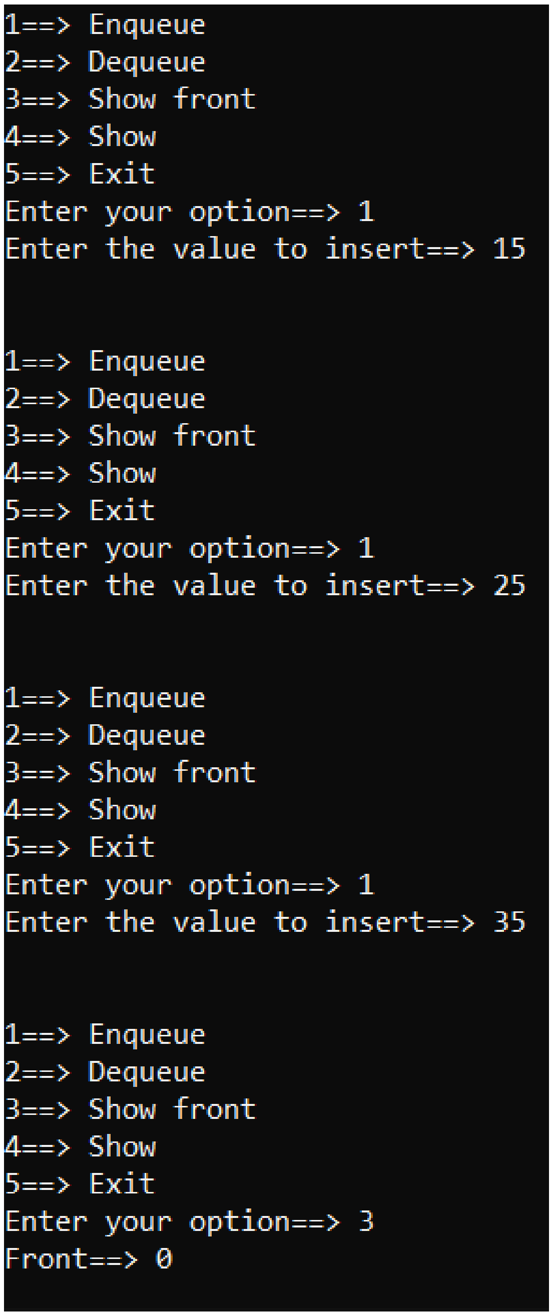
```
1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 2


1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 4
25

1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 2


1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 4
Stack is empty
```

8. Circular Queue using Array.

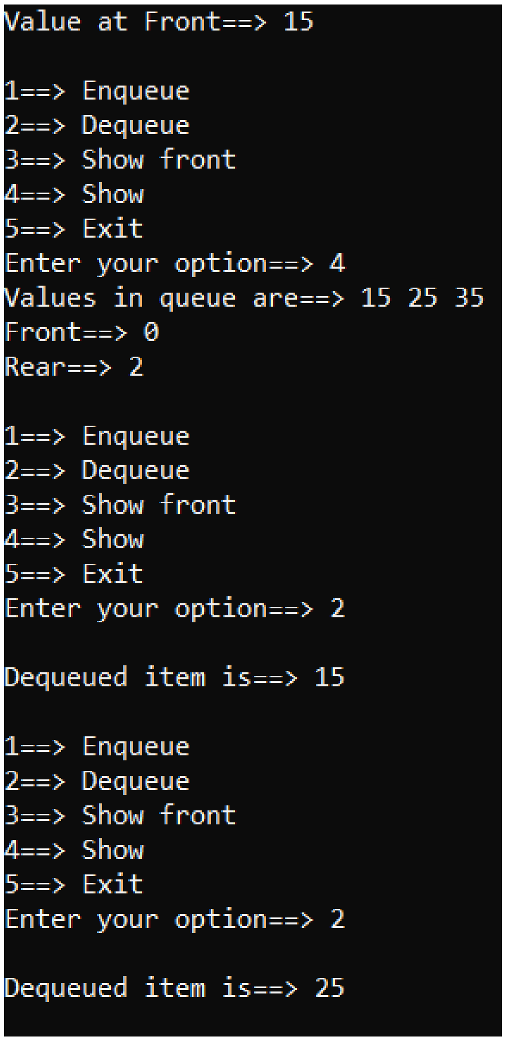
INPUT:

```
#include<stdio.h>
#include<conio.h>
int size=5;
int queue[5];
int front = -1;
int rear = -1;

void enqueue()
{
    int val;
    if(front==rear+1 || (front==0 && rear == size-1))
    {
        printf("Queue is full");
        return;
    }

    if(rear == -1 & front == -1 )
    {
        front += 1;
    }

    printf("Enter the value to insert==> ");
    scanf("%d", &val);
    rear = (rear + 1) % size;
    queue[rear] = val;
}

void dequeue()
{

    if(front==-1 && rear==-1)
    {
        printf("Queue is empty");
        return;
    }
    printf("\nDequeued item is==> %d", queue[front]);

    if(front == rear)
    {
        front=-1;
        rear=-1;
    }
    else front = (front + 1) % size;
}

void show()
{
    int i = 0;
    if(rear==-1)
    {
        printf("Queue is empty");
        return;
    }
    printf("Values in queue are==> ");

    for (i = front; i != rear; i = (i + 1) % size) {
      printf("%d ", queue[i]);
    }
    printf("%d ", queue[i]);
    printf("\nFront==> %d", front);
    printf("\nRear==> %d", rear);
}

void show_front()
{
    printf("Front==> %d", front);
    if(front!=-1) printf("\nValue at Front==> %d", queue[front]);
}

void main()
{
    int op = 0;
    while(op != 5)
    {
        printf("\n\n1==> Enqueue");
        printf("\n2==> Dequeue");
        printf("\n3==> Show front");
        printf("\n4==> Show");
        printf("\n5==> Exit");
        printf("\nEnter your option==> ");
        scanf("%d",&op);

        switch(op)
        {
            case 1:
                    enqueue();
                    break;

            case 2:
                    dequeue();
                    break;

            case 3:
                    show_front();
                    break;

            case 4:
                    show();
                    break;

            case 5:
                    printf("Exiting the program");
                    break;

            default: printf("Enter valid option");
                    break;
        }
    }

}
```

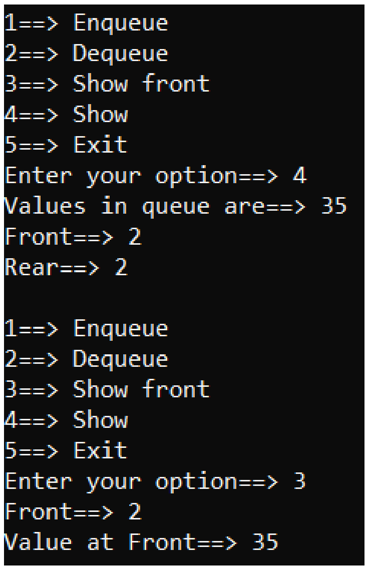
OUTPUT:



```
1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 1
Enter the value to insert==> 15


1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 1
Enter the value to insert==> 25


1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 1
Enter the value to insert==> 35


1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 3
Front==> 0
```



```
Value at Front==> 15

1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 4
Values in queue are==> 15 25 35
Front==> 0
Rear==> 2

1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 2

Dequeued item is==> 15

1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 2

Dequeued item is==> 25
```



```
1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 4
Values in queue are==> 35
Front==> 2
Rear==> 2

1==> Enqueue
2==> Dequeue
3==> Show front
4==> Show
5==> Exit
Enter your option==> 3
Front==> 2
Value at Front==> 35
```

9. Circular queue using Linked List.

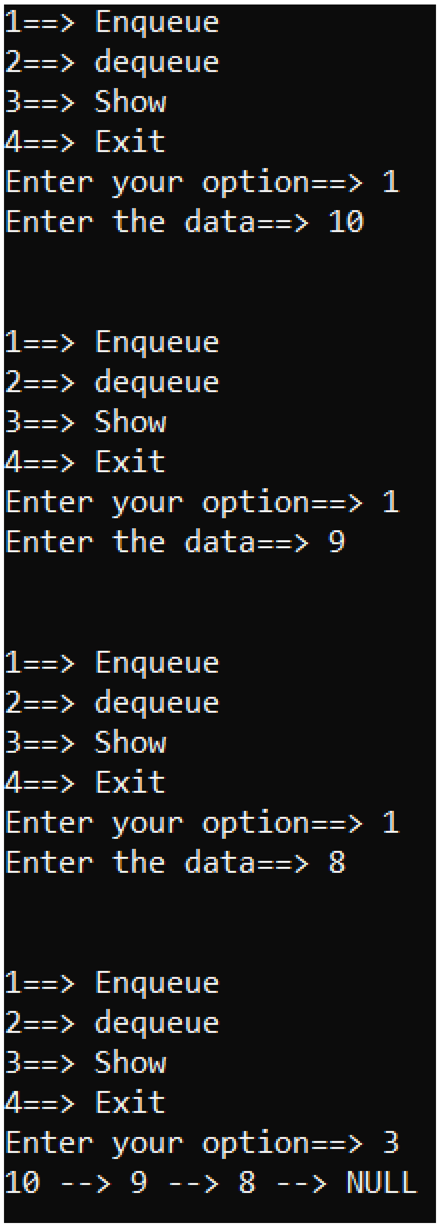
INPUT:

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

struct node
{
    int data;
    struct node *next;
}*new_node, *temp, *head, *tail;



void enqueue()
{
    int data;
    new_node = (struct node*)malloc(sizeof(struct node));
    printf("Enter the data==> ");
    scanf("%d",&data);
    new_node->data = data;

    if(head==NULL)
    {
        new_node->next = new_node;
        head = new_node;
        tail = new_node;
    }
    else
    {
        int data;
        temp = head;
        while(temp->next != head)
        {
            temp = temp->next;
        }
        new_node->next = head;
        temp->next = new_node;
        tail = new_node;
    }
```

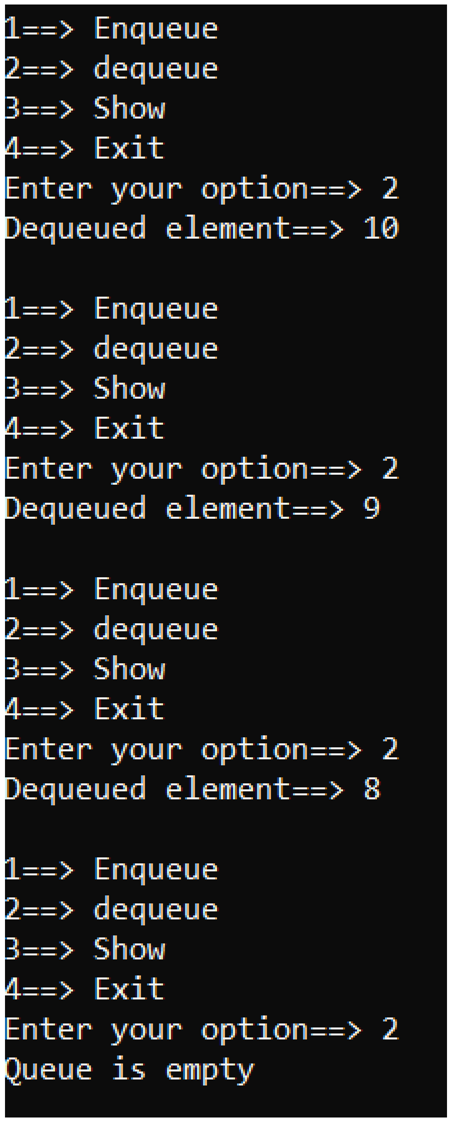
```c
}


void dequeue()
{
    int data;
    if(head==NULL)
    {
        printf("Queue is empty");
        return;
    }
    data = head->data;
    printf("Dequeued element==> %d", data);
    if(head->next == head)
    {
        free(head);
        tail = NULL;
        head = NULL;
        return;
    }
    temp = head->next;
    free(head);
    head = temp;
    tail->next = head;
}

void show()
{
    if(head==NULL)
    {
        printf("Queue is empty");
        return;
    }
    temp = head;

    do
    {
        printf("%d --> ", temp->data);
        temp = temp->next;
    }while(temp!=head);
    printf("NULL");
}
```

```c
void main()
{
    int op=0;
    while(op!=4)
    {
        printf("\n\n1==> Enqueue");
        printf("\n2==> dequeue");
        printf("\n3==> Show");
        printf("\n4==> Exit");
        printf("\nEnter your option==> ");
        scanf("%d",&op);
        switch (op)
        {
            case 1 : enqueue();
                break;

            case 2 : dequeue();
                break;

            case 3 : show();
                break;

            case 4 : printf("\nExiting!!!");
                break;

            default : printf("\nInvalid Input");
                break;
        }
    }
}
```

OUTPUT:



```
1==> Enqueue
2==> dequeue
3==> Show
4==> Exit
Enter your option==> 1
Enter the data==> 10


1==> Enqueue
2==> dequeue
3==> Show
4==> Exit
Enter your option==> 1
Enter the data==> 9


1==> Enqueue
2==> dequeue
3==> Show
4==> Exit
Enter your option==> 1
Enter the data==> 8


1==> Enqueue
2==> dequeue
3==> Show
4==> Exit
Enter your option==> 3
10 --> 9 --> 8 --> NULL
```



```
1==> Enqueue
2==> dequeue
3==> Show
4==> Exit
Enter your option==> 2
Dequeued element==> 10

1==> Enqueue
2==> dequeue
3==> Show
4==> Exit
Enter your option==> 2
Dequeued element==> 9

1==> Enqueue
2==> dequeue
3==> Show
4==> Exit
Enter your option==> 2
Dequeued element==> 8

1==> Enqueue
2==> dequeue
3==> Show
4==> Exit
Enter your option==> 2
Queue is empty
```

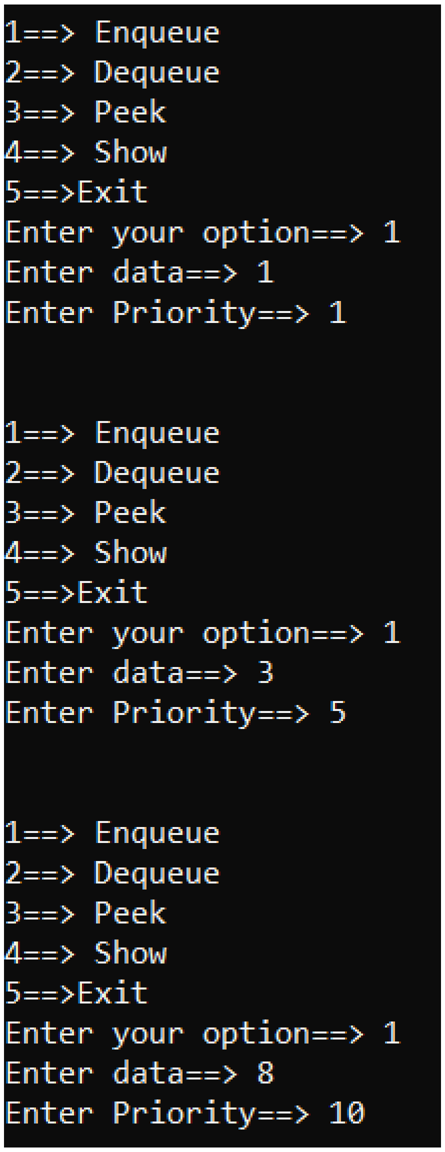
10. Priority Queue using Array.

INPUT:

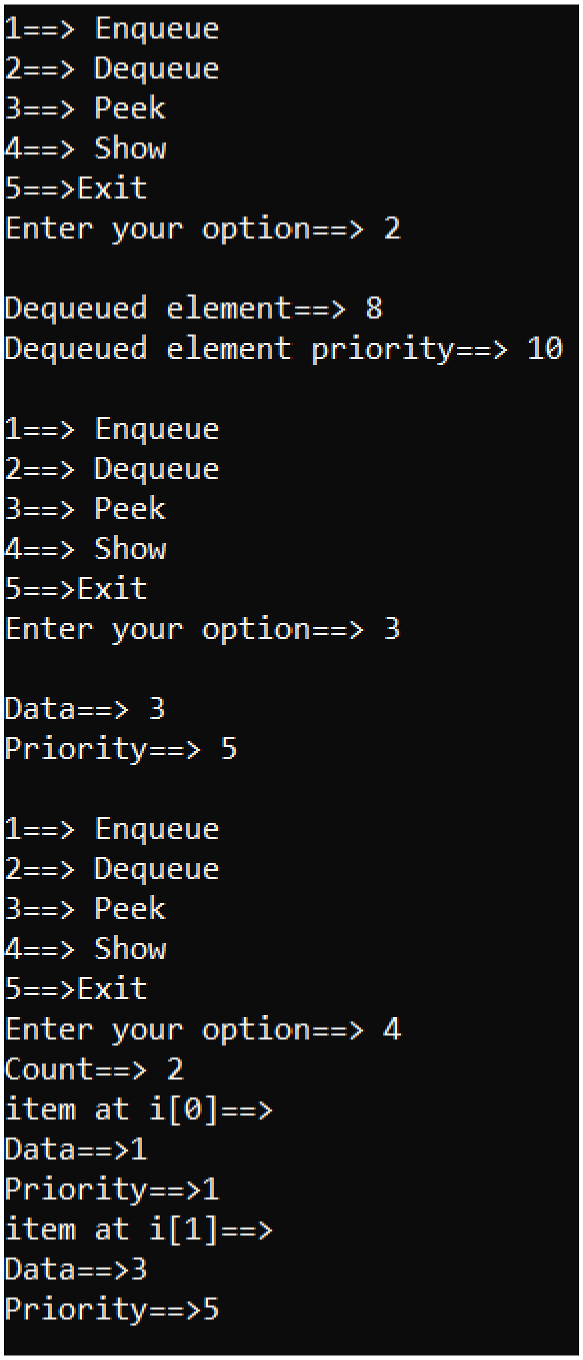
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define SIZE 5

int count = 0;

struct item
{
    int data;
    int priority;
};

struct item pqueue[SIZE];

void enqueue()
{
    int data;
    int priority;
    if(SIZE == count)
    {
        printf("Overflow");
        return;
    }
    printf("Enter data==> ");
    scanf("%d", &data);
    printf("Enter Priority==> ");
    scanf("%d", &priority);
    pqueue[count].data = data;
    pqueue[count].priority = priority;
    count++;
}

int peek()
{
    int i;
    int highestPriority = INT_MIN;
    int index = -1;

    for(i=0;i<count;i++)
    {
        if(highestPriority==pqueue[i].priority && index > -1 &&
 pqueue[index].data < pqueue[i].data)
        {
            highestPriority = pqueue[i].priority;
            index = i;
        }
        else if(highestPriority<pqueue[i].priority)
        {
            highestPriority = pqueue[i].priority;
            index = i;
        }
    }

    return index;
}

void dequeue()
{
    int index = peek();
    if(index == -1)
    {
        printf("Underflow");
        return;
    }
    printf("\nDequeued element==> %d",pqueue[index].data);
    printf("\nDequeued element priority==>
%d",pqueue[index].priority);
    for(int i = index;i<count;i++)
    {
        pqueue[i] = pqueue[i+1];
    }
    count--;
}
```

```
void showPeek()
{
    int index = peek();
    if(index==-1)
    {
        printf("\nUnderflow");
        return;
    }
    printf("\nData==> %d", pqueue[index].data);
    printf("\nPriority==> %d", pqueue[index].priority);
}

void show()
{
    int i=0;
    printf("Count==> %d", count);
    for(i=0;i<count;i++)
    {
        printf("\nitem at i[%d]==> ", i);
        printf("\nData==>%d", pqueue[i].data);
        printf("\nPriority==>%d", pqueue[i].priority);

    }
}


void main()
{
    int op = 0;
    int data;
    while(op != 5)
    {
        printf("\n\n1==> Enqueue");
        printf("\n2==> Dequeue");
        printf("\n3==> Peek");
        printf("\n4==> Show");
        printf("\n5==>Exit");
        printf("\nEnter your option==> ");
        scanf("%d",&op);

        switch(op)
        {
            case 1:
                    enqueue();
                    break;

            case 2:
                    dequeue();
                    break;

            case 3:
                    showPeek();
                    break;

            case 4:
                    show();
                    break;

            case 5:
                    printf("Exiting!!!");
                    break;

            default:
                    printf("Invalid Input");
                    break;
        }
    }
}
```

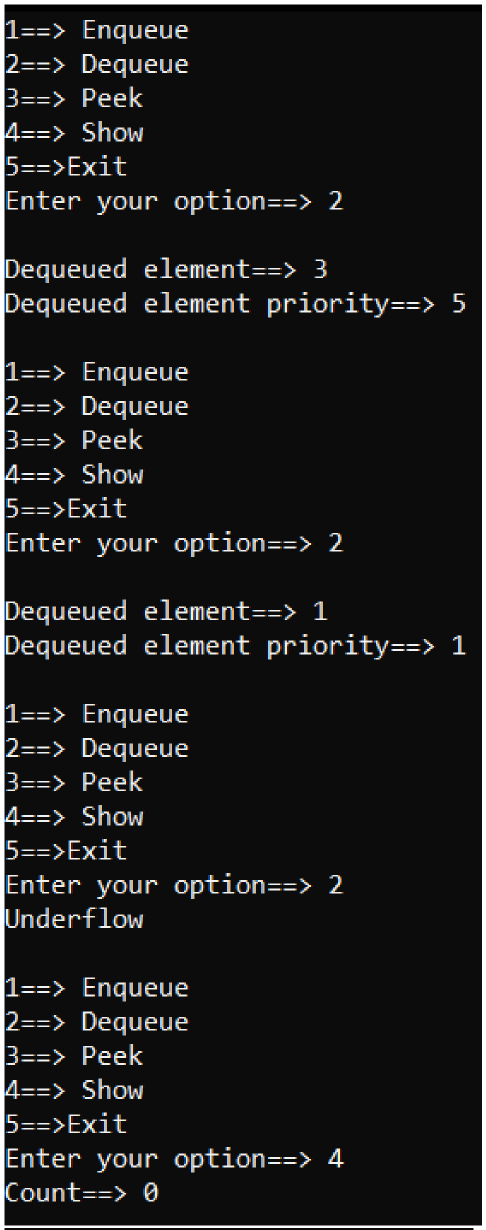
OUTPUT:



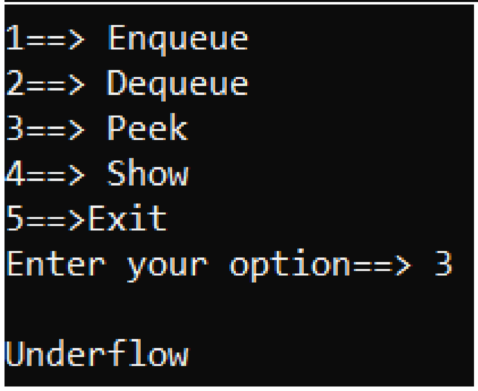
```
1==> Enqueue
2==> Dequeue
3==> Peek
4==> Show
5==>Exit
Enter your option==> 1
Enter data==> 1
Enter Priority==> 1


1==> Enqueue
2==> Dequeue
3==> Peek
4==> Show
5==>Exit
Enter your option==> 1
Enter data==> 3
Enter Priority==> 5


1==> Enqueue
2==> Dequeue
3==> Peek
4==> Show
5==>Exit
Enter your option==> 1
Enter data==> 8
Enter Priority==> 10
```



```
1==> Enqueue
2==> Dequeue
3==> Peek
4==> Show
5==>Exit
Enter your option==> 2

Dequeued element==> 8
Dequeued element priority==> 10

1==> Enqueue
2==> Dequeue
3==> Peek
4==> Show
5==>Exit
Enter your option==> 3

Data==> 3
Priority==> 5

1==> Enqueue
2==> Dequeue
3==> Peek
4==> Show
5==>Exit
Enter your option==> 4
Count==> 2
item at i[0]==>
Data==>1
Priority==>1
item at i[1]==>
Data==>3
Priority==>5
```



```
1==> Enqueue
2==> Dequeue
3==> Peek
4==> Show
5==>Exit
Enter your option==> 2

Dequeued element==> 3
Dequeued element priority==> 5

1==> Enqueue
2==> Dequeue
3==> Peek
4==> Show
5==>Exit
Enter your option==> 2

Dequeued element==> 1
Dequeued element priority==> 1

1==> Enqueue
2==> Dequeue
3==> Peek
4==> Show
5==>Exit
Enter your option==> 2
Underflow

1==> Enqueue
2==> Dequeue
3==> Peek
4==> Show
5==>Exit
Enter your option==> 4
Count==> 0
```



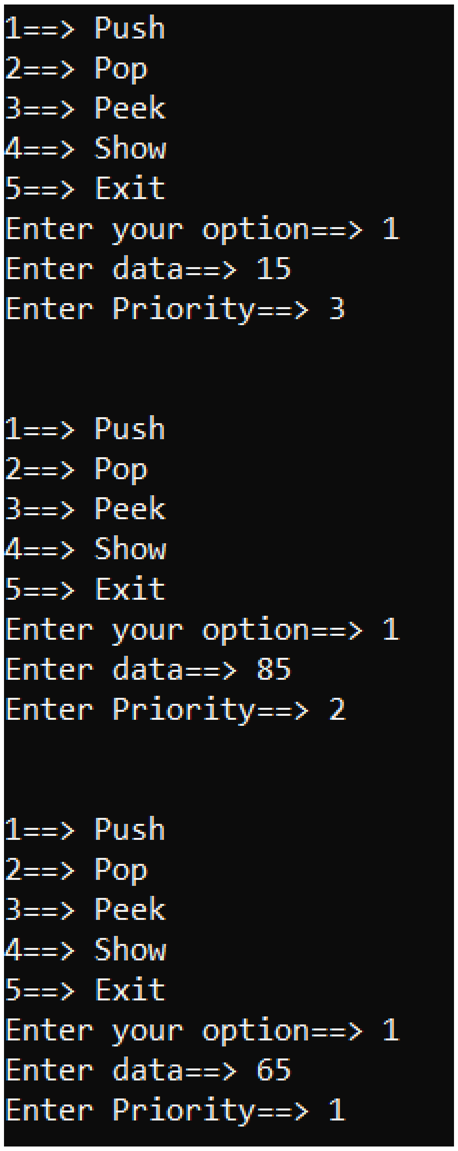
```
1==> Enqueue
2==> Dequeue
3==> Peek
4==> Show
5==>Exit
Enter your option==> 3

Underflow
```

11. Priority Queue using Linked List.

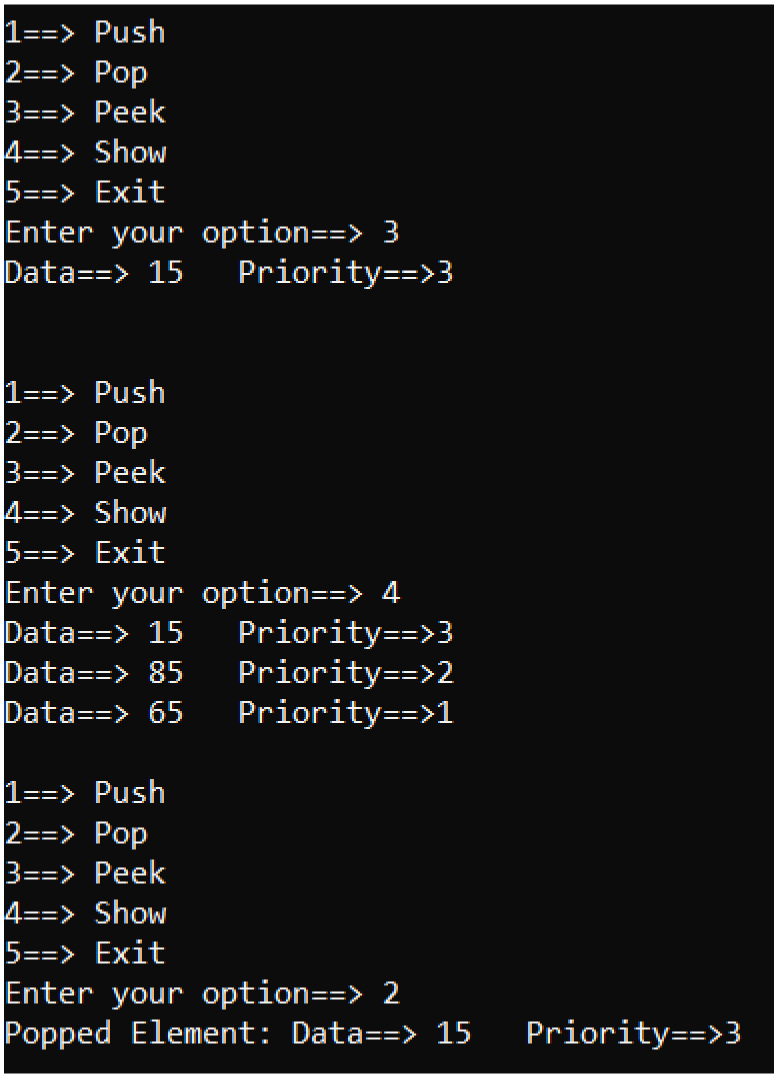
INPUT:

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

struct node
{
    int data;
    int priority;
    struct node *next;
}*new_node, *temp, *head;


void push()
{
    int data, priority;
    struct node *ptr, *preptr;
    printf("Enter data==> ");
    scanf("%d", &data);
    printf("Enter Priority==> ");
    scanf("%d", &priority);
    new_node = (struct node*)malloc(sizeof(struct node));
    new_node->data = data;
    new_node->priority = priority;
    if(head == NULL)
    {
        new_node->next = NULL;
        head = new_node;
        return;
    }
    ptr = head;
    while(ptr != NULL && ptr->priority>priority)
    {
        preptr = ptr;
        ptr = ptr->next;
    }
    if(ptr==head)
    {
        new_node->next = head;
        head = new_node;
        return;
    }
    preptr->next = new_node;
    new_node->next = ptr;
}

void pop()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("Stack Underflow");
        return;
    }
    ptr = head;
    head = head->next;
    printf("Popped Element: Data==> %d   Priority==>%d\n", ptr->data, 
ptr->priority);
    free(ptr);
}

void peek()
{
    printf("Data==> %d   Priority==>%d\n", head->data, head-
>priority);
}

void show()
{
    struct node *ptr;
    if(head==NULL)
    {
        printf("Stack Underflow");
    }
    ptr = head;
    while(ptr->next != NULL)
    {
        printf("Data==> %d   Priority==>%d\n", ptr->data, ptr-
>priority);
        ptr = ptr->next;
    }
    printf("Data==> %d   Priority==>%d", ptr->data, ptr->priority);
}

void main()
{
    int op=0;
    while(op!=5)
    {
        printf("\n\n1==> Push");
        printf("\n2==> Pop");
        printf("\n3==> Peek");
        printf("\n4==> Show");
        printf("\n5==> Exit");
        printf("\nEnter your option==> ");
        scanf("%d",&op);
        switch (op)
        {
            case 1 : push();
                break;

            case 2 : pop();
                break;

            case 3 : peek();
                break;

            case 4 : show();
                break;

            case 5 : printf("\nExiting!!!");
                break;

            default : printf("\nInvalid Input");
                break;
            }
        }
}
```

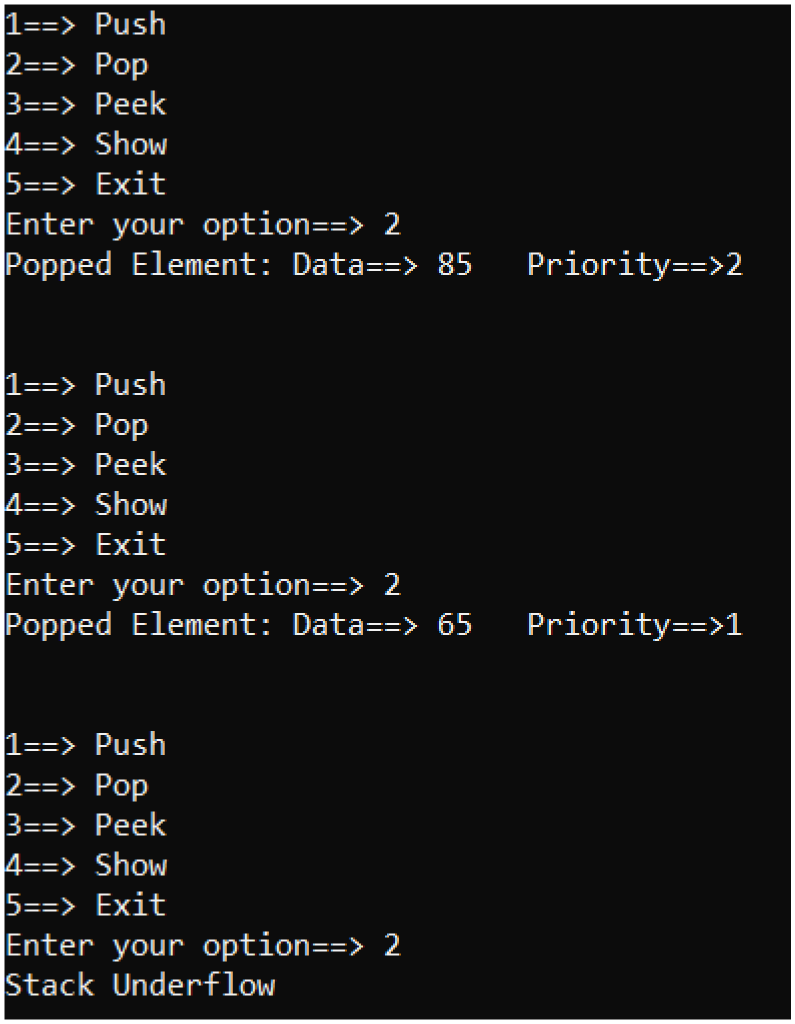
OUTPUT:



```
1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 1
Enter data==> 15
Enter Priority==> 3


1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 1
Enter data==> 85
Enter Priority==> 2


1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 1
Enter data==> 65
Enter Priority==> 1
```



```
1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 3
Data==> 15   Priority==>3


1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 4
Data==> 15   Priority==>3
Data==> 85   Priority==>2
Data==> 65   Priority==>1

1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 2
Popped Element: Data==> 15   Priority==>3
```



```
1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 2
Popped Element: Data==> 85   Priority==>2


1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 2
Popped Element: Data==> 65   Priority==>1


1==> Push
2==> Pop
3==> Peek
4==> Show
5==> Exit
Enter your option==> 2
Stack Underflow
```

DS ASSIGNMENT

Roll No: 40 **Name: Akshit Trivedi**
Class: MCA-1 **Year: 2021-22**

12. Binary Search Tree.

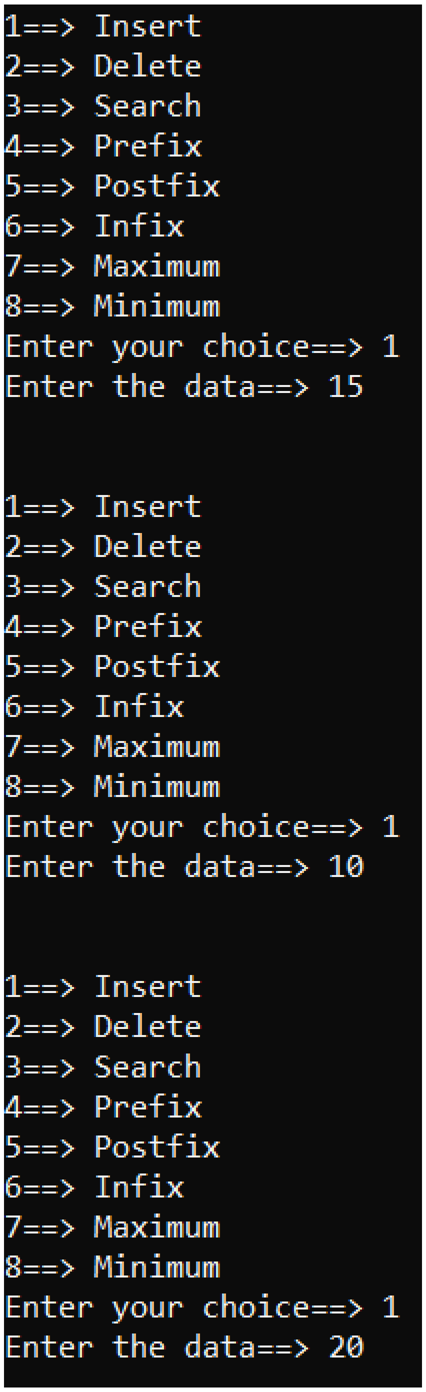
INPUT:

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

struct Node
{
    int data;
    struct Node *leftChild;
    struct Node *rightChild;
}*root, *new_node;

void preorder(struct Node* node)
{
    if(node == NULL) return;
    printf("%d " ,node->data);
    preorder(node->leftChild);
    preorder(node->rightChild);
}

void postorder(struct Node* node)
{
    if(node == NULL) return;
    postorder(node->leftChild);
    postorder(node->rightChild);
    printf("%d " ,node->data);
}

void inorder(struct Node* node)
{
    if(node == NULL) return;
    inorder(node->leftChild);
    printf("%d " ,node->data);
    inorder(node->rightChild);

}
```

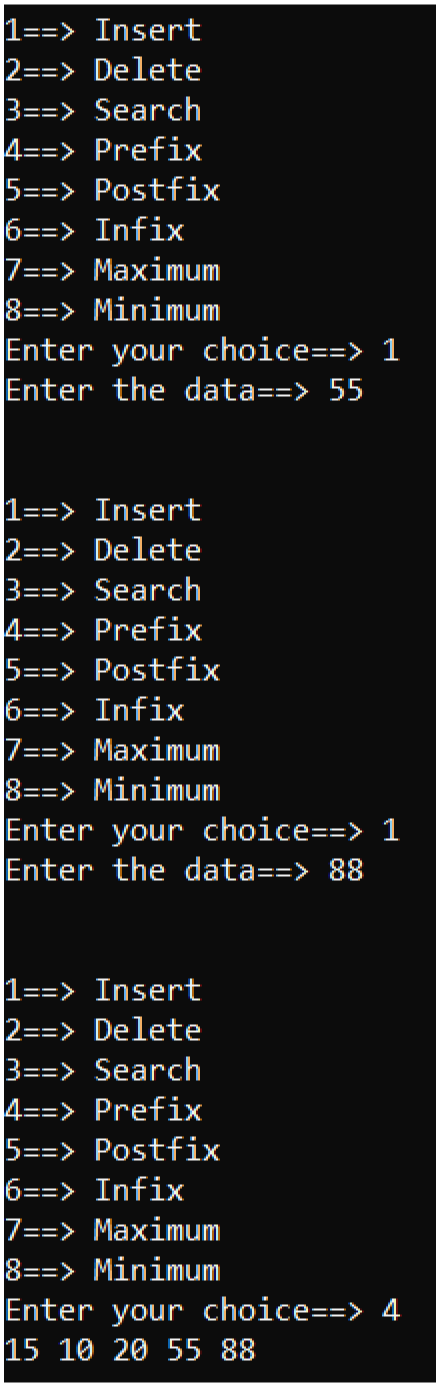
```
void find_max()
{
    if(root == NULL)
    {
        printf("Tree is empty");
    }
    new_node = root;
    while(new_node->rightChild != NULL)
    {
        new_node = new_node->rightChild;
    }
    printf("Maximum==> %d", new_node->data);
}


void find_min()
{
    if(root == NULL)
    {
        printf("Tree is empty");
    }

    new_node = root;

    while(new_node->leftChild != NULL)
    {
        new_node = new_node->leftChild;
    }

    printf("Minimum==> %d", new_node->data);
}


struct Node* createNewNode(int data)
{
    new_node = (struct Node*)malloc(sizeof(struct Node));
    new_node->data = data;
    new_node->leftChild = NULL;
    new_node->rightChild = NULL;
    if(root == NULL)
    {
        root = new_node;
        return root;
    }
    return new_node;
}

struct Node* insert(struct Node* node, int data)
{
    if (node==NULL)
         return createNewNode(data);

    if(data < node->data)
        node->leftChild = insert(node->leftChild, data);

    else if(data > node->data)
        node->rightChild = insert(node->rightChild, data);

    return node;

}


void search(int data)
{
    int flag = 0;
    new_node = root;
    while(new_node != NULL)
    {
        if(data == new_node->data)
        {
           flag=1;
           break;
        }
        else if(data < new_node->data)
        {
            new_node = new_node->leftChild;
        }
        else
        {
            new_node = new_node->rightChild;
        }
    }

    if(flag == 0) printf("%d does not exist in the tree", data);
    else printf("%d exist in the tree", data);
}


void main()
{
    int op = 0;
    int data;
    while(op != 12)
    {
        printf("\n\n1==> Insert");
        printf("\n2==> Delete");
        printf("\n3==> Search");
        printf("\n4==> Prefix");
        printf("\n5==> Postfix");
        printf("\n6==> Infix");
        printf("\n7==> Maximum");
        printf("\n8==> Minimum");
        printf("\nEnter your choice==> ");
        scanf("%d",&op);

        switch(op)
        {
            case 1:
                    printf("Enter the data==> ");
                    scanf("%d", &data);
                    insert(root, data);
                    break;


            case 2:
                    printf("Enter the data==> ");
                    scanf("%d", &data);
                    break;

            case 3:
                    printf("Enter the data==> ");
                    scanf("%d", &data);
                    search(data);
                    break;

            case 4:
                    preorder(root);
                    break;

            case 5:
                    postorder(root);
                    break;

            case 6:
                    inorder(root);
                    break;

            case 7:
                    find_max();
                    break;

            case 8:
                    find_min();
                    break;

            case 12:
                    printf("Exiting!!!");
                    break;

            default:
                    printf("Please enter an valid input");
                    break;
        }
    }
```

```
}
```

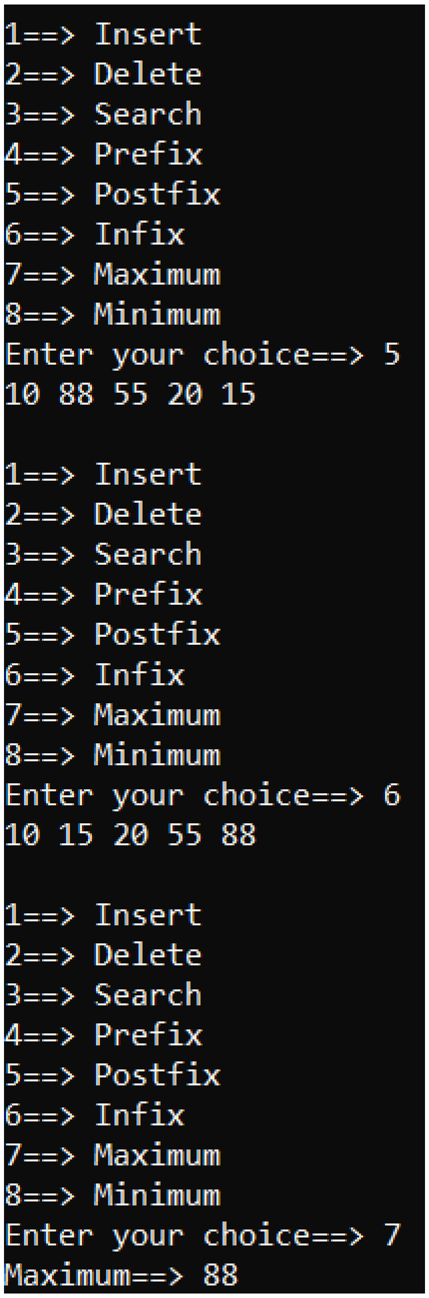
OUTPUT:



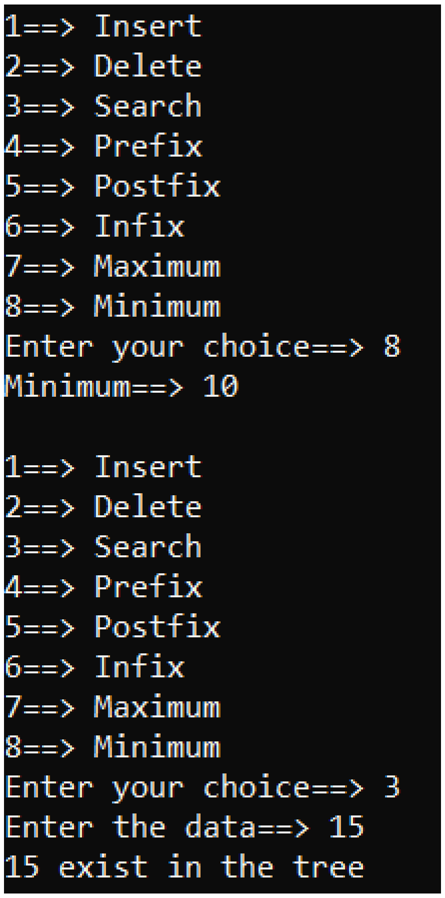
```
1==> Insert
2==> Delete
3==> Search
4==> Prefix
5==> Postfix
6==> Infix
7==> Maximum
8==> Minimum
Enter your choice==> 1
Enter the data==> 15


1==> Insert
2==> Delete
3==> Search
4==> Prefix
5==> Postfix
6==> Infix
7==> Maximum
8==> Minimum
Enter your choice==> 1
Enter the data==> 10


1==> Insert
2==> Delete
3==> Search
4==> Prefix
5==> Postfix
6==> Infix
7==> Maximum
8==> Minimum
Enter your choice==> 1
Enter the data==> 20
```



```
1==> Insert
2==> Delete
3==> Search
4==> Prefix
5==> Postfix
6==> Infix
7==> Maximum
8==> Minimum
Enter your choice==> 1
Enter the data==> 55


1==> Insert
2==> Delete
3==> Search
4==> Prefix
5==> Postfix
6==> Infix
7==> Maximum
8==> Minimum
Enter your choice==> 1
Enter the data==> 88


1==> Insert
2==> Delete
3==> Search
4==> Prefix
5==> Postfix
6==> Infix
7==> Maximum
8==> Minimum
Enter your choice==> 4
15 10 20 55 88
```



```
1==> Insert
2==> Delete
3==> Search
4==> Prefix
5==> Postfix
6==> Infix
7==> Maximum
8==> Minimum
Enter your choice==> 5
10 88 55 20 15

1==> Insert
2==> Delete
3==> Search
4==> Prefix
5==> Postfix
6==> Infix
7==> Maximum
8==> Minimum
Enter your choice==> 6
10 15 20 55 88

1==> Insert
2==> Delete
3==> Search
4==> Prefix
5==> Postfix
6==> Infix
7==> Maximum
8==> Minimum
Enter your choice==> 7
Maximum==> 88
```



```
1==> Insert
2==> Delete
3==> Search
4==> Prefix
5==> Postfix
6==> Infix
7==> Maximum
8==> Minimum
Enter your choice==> 8
Minimum==> 10

1==> Insert
2==> Delete
3==> Search
4==> Prefix
5==> Postfix
6==> Infix
7==> Maximum
8==> Minimum
Enter your choice==> 3
Enter the data==> 15
15 exist in the tree
```

DS ASSIGNMENT

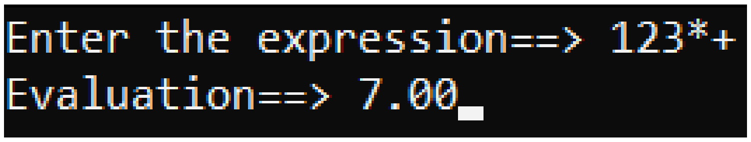
Roll No: 40 **Name: Akshit Trivedi**
Class: MCA-1 **Year: 2021-22**

13. Postfix Evaluation.

INPUT:

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<conio.h>
#include<stdlib.h>

float stack[100];
int max = 100;
int top=-1;

void push(int item)
{
    stack[++top] = item;
}

float pop()
{
    if(top<0)
    {
        printf("\nStack underflow");
        return INT_MIN;
    }
    return stack[top--];
}


void evaluate_postfix(char* exp)
{
    int i;
    int len = strlen(exp);

    for(i=0;i<len;i++)
    {
        if(isdigit(exp[i]))
        {
            push(exp[i] - '0');
        }
        else
        {
            float a = pop();
            float b = pop();
            if(a == INT_MIN || b == INT_MIN)
            {
                printf("Wrong expression!!!");
                return;
            }
            switch (exp[i])
            {
                case '+':
                            push(b + a);
                            break;
                case '-':   push(b - a);
                            break;
                case '*':   push(b * a);
                            break;
                case '/':   push(b / a);
                            break;
            }
        }
    }
    printf("Evaluation==> %.2f", pop());
}

void main()
{
    char exp[100];
    printf("Enter the expression==> ");
    scanf("%s",&exp);
    evaluate_postfix(exp);
    getch();
}
```

OUTPUT:



```
Enter the expression==> 123*+
Evaluation==> 7.00_
```