B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Sep-Dec 2020

Department of Computer Science and Engineering

DATA STRUCTURE LAB REPORT
(19CS3PCDST)

Submitted by

Avirath Manohar Hegde

USN: - 1BM19CS195

3RD SEMESTER

D – SECTION

Lab In charge

Madhavi R. P.

Associate Professor

Dept. Of Computer Science

LAB 1

Q. Write a program to simulate the working of stack using an array with the following

A) Push
B) Pop
C) Display

The Program should print appropriate messages for stack overflow and stack underflow.

PROGRAM

```
#include<stdio.h>
#include<stdlib.h
> #define SIZE 10

void push(int);
void pop();
void display();
int S[SIZE],top=-
1; int main()
{
        int x,a;
        while(1)
        {
                printf("[1]PUSH\n[2]POP\n[3]DISPLAY\n[4]EXIT\n");
                printf("..................................");
                printf("\nEnter  Your  Choice:
                "); scanf("%d",&a);

                switch(a)
```

```
                {
                        case 1: printf("\nEnter the value to be inserted: ");
                                scanf("%d",&x);
                                push(x);
                                break;
                        case 2: pop();
                                break;
                        case 3: display();
                                break;
                        case 4: exit(0);
                        default: printf("\nWrong Selection!!!Select Again!!!\n");
                }
        }
}

void push(int x)
{
        if(top==SIZE-1)
                printf("\nStack is Full!!!\n\n");
        else
        {
                top++;
                S[top]=x
                ;
        }       printf("\nElement successsfuly Inserted!!!\n\n");
}

void pop()
{
```

```
	if(top==-1)
		printf("\nStack is Empty!!!\n\n");
	else
	{
		printf("\nDeleted: %d\n\n",S[top]);
		top--;
	}
}

void display()
{
	if (top==-1)
		printf("\nStack is Empty!!!\n\n");
	else
	{
		int i;
		printf("\nStack Elements are:\n");
		for(i=top;i>=0;i--)
			printf("%d\n",S[i])
			; printf("\n\n");
	}
}
```

OUTPUT

```
[2]POP
[3]DISPLAY
[4]EXIT
----------------------------
Enter Your Choice: 1

Enter the value to be inserted: 3

Element successsfuly Inserted!!!

[1]PUSH
[2]POP
[3]DISPLAY
[4]EXIT
----------------------------
Enter Your Choice: 3

Stack Elements are:
3
5


[1]PUSH
[2]POP
[3]DISPLAY
[4]EXIT
----------------------------
Enter Your Choice:
```

```
[1]PUSH
[2]POP
[3]DISPLAY
[4]EXIT
----------------------------
Enter Your Choice: 3

Stack Elements are:
3
5


[1]PUSH
[2]POP
[3]DISPLAY
[4]EXIT
----------------------------
Enter Your Choice: 2

Deleted: 3

[1]PUSH
[2]POP
[3]DISPLAY
[4]EXIT
----------------------------
Enter Your Choice:
```

LAB 2

Q. WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators +(plus), -(minus), *(multiply) and /(divide)

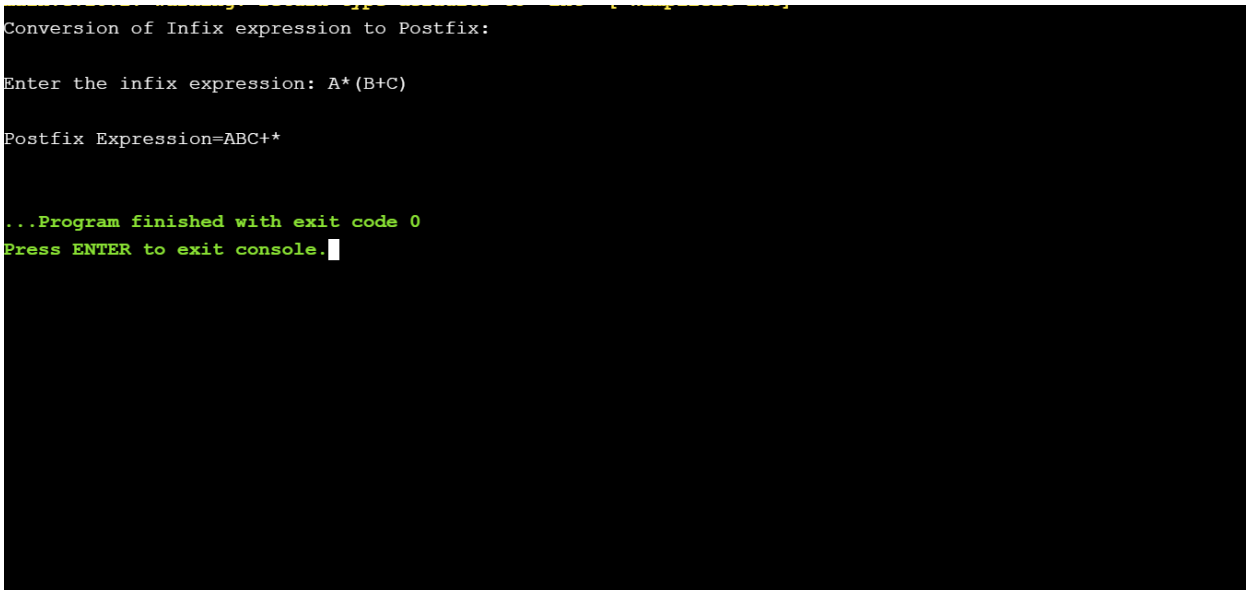
PROGRAM

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


#define SIZE 50


char stack[SIZE];
int top=-1;


push(char a)
{
        stack[++top]=a;
}


char pop()
{
        return(stack[top--]);
}


int operator(char symbol)
{
        if(symbol=='^')
        {
                return(3);
        }
        else if(symbol=='*'||symbol=='/')
        {
                return(2);
        }
        else if(symbol=='+'||symbol=='-')
```

```
	{
		return (1);
	}
	else
	{
		return(0);
	}
}


int main()
{
	char infix[50],postfix[50],x,elem;
	int i=0,k=0;

	printf("Conversion of Infix expression to Postfix:\n\n");
	printf("Enter the infix expression: ");
	scanf("%s",infix);

	push('#');

	while((x=infix[i++])!='\0')
	{
		if(x=='(')push(x);
		else
			if(isalnum(x))postfix[k++]=x;
			else
				if(x==')')
				{
					while(stack[top]!='(')
					{
						postfix[k++]=pop();
					}
					elem=pop();
				}
				else
				{
					while(operator(stack[top])>=operator(x))
					{
						postfix[k++]=pop();
					}
					push(x);
				}
	}
	while(stack[top]!='#')
		{
			postfix[k++]=pop();
		}
	postfix[k]='\0';
	printf("\nPostfix Expression=%s\n",postfix);
```

```
        return 0;
    }
```

OUTPUT



LAB 3

Q. WAP to simulate the working of a queue of integers using an array. Provide the following operations

a) Insert

b) Delete

c) Display

The program should print appropriate messages for queue empty and queue overflow conditions

PROGRAM

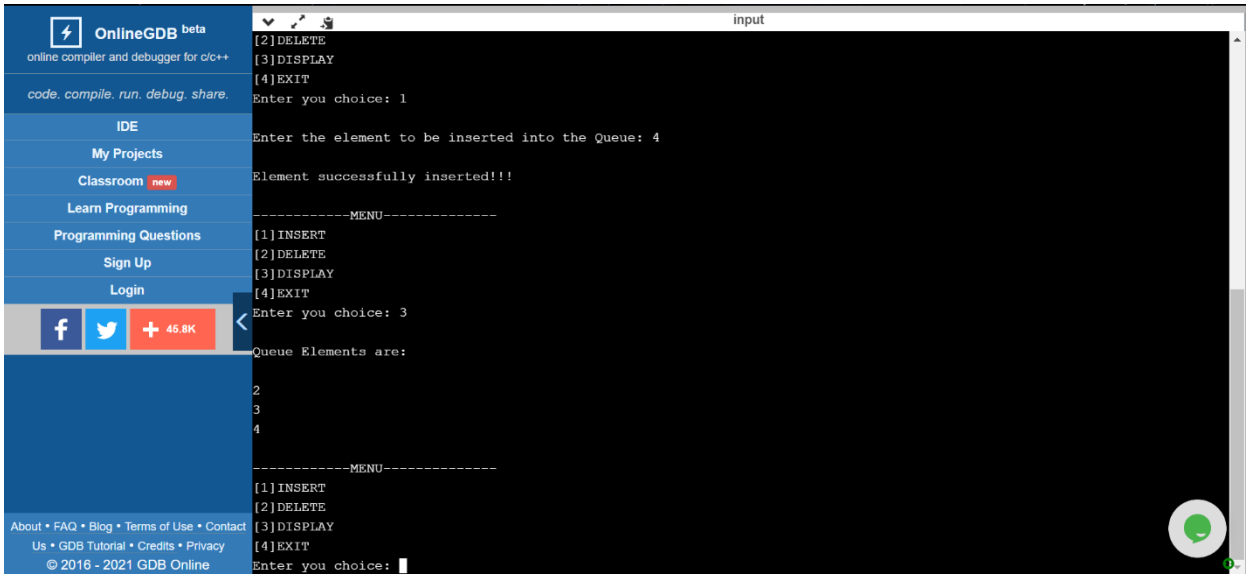
```
#include<stdio.h>
#include<stdlib.h>
#define MAX 50
int Q[MAX];
int front = -1;
int rear = -1;

void insert()
{
        int elem;
        if(rear==MAX-
        1)
                printf("\nQueue Overflow!!!\n");
        else
        {
                if(front==-1)
                        front=0;
                printf("\nEnter the element to be inserted into the Queue: ");
                scanf("%d",&elem);
                Q[++rear]=elem;
```

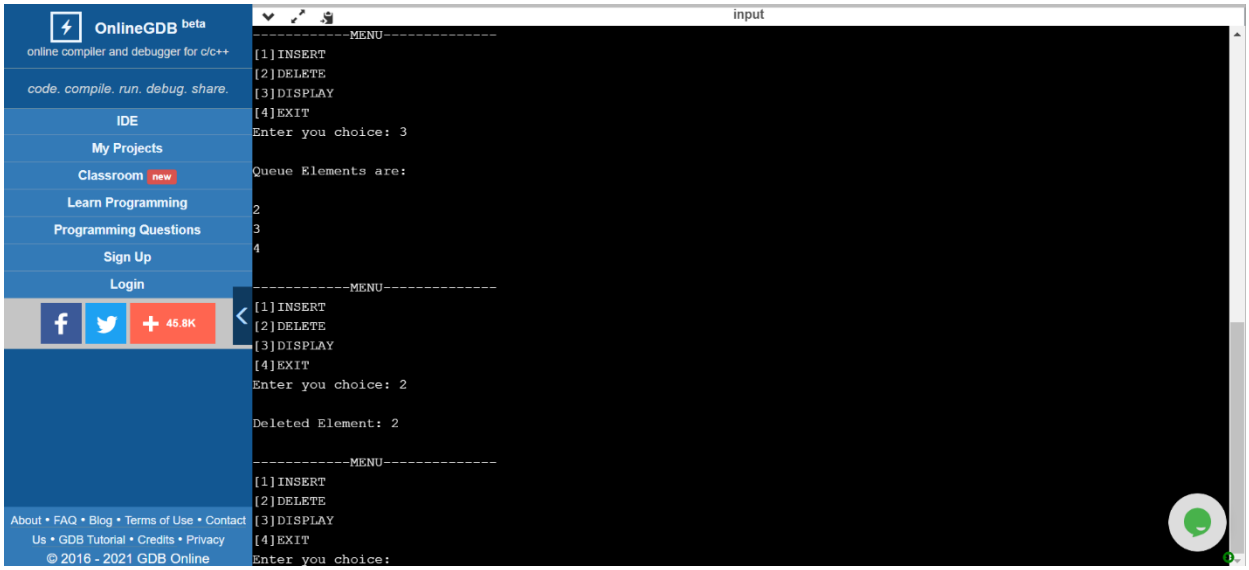
```
                printf("\nElement successfully inserted!!!\n");
        }
}

void delete()
{
        if(front==-1||front>rear) printf("\nQueue
                underflow!!!\n");
        else
        {
                printf("\nDeleted Element: %d\n",Q[front++]);
                if(front>rear)
                {
                        front=-1;
                        rear=-1;
                }
        }
}

void display()
{
        int i;
        if(front==-
        1)
                printf("\nQueue is empty!!!\n");
        else
        {
                printf("\nQueue Elements are: \n");
                for(i=front;i<=rear;i++)
                        printf("\n%d",Q[i]);
```

```
                printf("\n");
        }
}
int main()
{
        int choice;
        while(1)
        {
                printf("\n.....MENU.....");
                printf("\n[1]INSERT\n[2]DELETE\n[3]DISPLAY\n[4]E
                XIT");
                printf("\nEnter you choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:
                                insert();
                                break;
                        case 2:
                                delete();
                                break;
                        case 3:
                                display()
                                ; break;
                        case 4:
                                exit(1);
                        default:
                                printf("\nInvalid choice!!!\n");
                }
        }
```

```
    return 0;
}
```

OUTPUT





LAB 4

Q. WAP to simulate the working of a circular queue of integers using an array. Provide the following operations.

a) Insert

b) Delete

c) Display

The program should print appropriate messages for queue empty and queue overflow conditions

PROGRAM

```
#include<stdio.h>

#include<stdlib.h

> #define size 5
```

```
int Q[size];
int front=-1,rear=-1;

int isfull()
{
	if(front==rear+1||(front==0&&rear==size-1))
	{
		return 1;
	}
	return 0;
}

int isempty()
{
	if(front==-1)
	{
		return 1;
	}
	return 0;
}

void enqueue()
{
	int elem;
	if(isfull()
	)
	{
		printf("\nQueue Overflow!!!\n");
	}
	else
```

```
	{
		if(front==-1)
		{
			front=0;
		}
		printf("\nEnter the element to be inserted into the Queue: ");
		scanf("%d",&elem);
		rear =
		(rear+1)%size;
		Q[rear]=elem;
		printf("\nInserted-->%d\n",elem);
	}
}

void dequeue()
{
	int elem;
	if(isempty()
	)
	{
		printf("\nQueue Underflow!!!\n");
	}
	else
	{
		elem=Q[front];
		if(front==rear)
		{
			front=-1;
			rear=-1;
		}
		else
```

```
                {
                        front=(front+1)%size;
                }
                printf("\nDeleted Element-->%d\n",elem);
        }
}

void display()
{
        int i;
        if(isempty()
        )
        {
                printf("\nQueue is Empty!!!Enter some Elements!!!\n");
        }
        else
        {
                printf("\nFront--> %d",front);
                printf("\nQueue Elements--> \n");
                for(i=front;i!=rear;i=(i+1)%size)
                {
                        printf("%d\n",Q[i]);
                }
                printf("%d",Q[i]);
                printf("\nRear--> %d\n",rear);
        }
}
int main()
{
        int choice;
```

```
        while(1)
        {
                printf("\n.....MENU.....");
                printf("\n[1]INSERT\n[2]DELETE\n[3]DISPLAY\n[4]E
                XIT");
                printf("\nEnter you choice: ");
                scanf("%d",&choice);

                switch(choice)
                {
                        case 1:
                                enqueue();
                                break;
                        case 2:
                                dequeue();
                                break;
                        case 3:
                                display()
                                ; break;
                        case 4:
                                exit(1);
                        default:
                                printf("\nInvalid choice!!!\n");
                }
        }
        return 0;
}
```

OUTPUT

```
-----------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 1

Enter the element to be inserted into the Queue: 5

Inserted-->5

-----------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 1

Queue OverFlow!!!

-----------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 3

Front--> 0
Queue Elements-->
1
2
3
4
5
Rear--> 4

          MENU
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice:
2

Deleted Element-->1

          MENU
[1]INSERT
[2]DELETE
```

```
-----------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 1

Enter the element to be inserted into the Queue: 1

Inserted-->1

-----------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 1

Enter the element to be inserted into the Queue: 2

Inserted-->2

-----------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 1

Enter the element to be inserted into the Queue: 3

Inserted-->3

-----------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 1

Enter the element to be inserted into the Queue: 4

Inserted-->4

-----------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
```


[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 3
Front--> 1
Queue Elements-->
2
3
4
5
Rear--> 4
------------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 2
Deleted Element-->2
------------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 3
Front--> 2
Queue Elements-->
3
4
5
Rear--> 4
------------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 1
Enter the element to be inserted into the Queue: 3
Inserted-->3


[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 3
Front--> 2
Queue Elements-->
3
4
5
3
Rear--> 0
------------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 2
Deleted Element-->3
------------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 3
Front--> 3
Queue Elements-->
4
5
3
Rear--> 0
------------MENU-------------
[1]INSERT
[2]DELETE
[3]DISPLAY
[4]EXIT
Enter you choice: 4

(program exited with code: 1)

LAB 5 & LAB 6

Q. WAP to Implement Singly Linked List with following operations

a) Create a linked list.

b) Insertion of a node at first position, at any position and at end of list.

c) Deletion of first element, specified element and last element in the list.

d) Display the contents of the linked list.

PROGRAM

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

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

void insert_front()
{
        struct node *ptr;
        int new_data;
        ptr = (struct node *)malloc(sizeof(struct node));

        if(ptr == NULL)
        {
```

```
            printf("\nOVERFLOW!!!");
        }
        else
        {
            printf("\nEnter the Value to be inserted:");
            scanf("%d",&new_data);
            ptr->data = new_data;
            ptr->next = head; head
            = ptr;
            printf("\nNODE INSERTED AT THE FRONT\n");
        }
}

void insert_end()
{
        struct node *ptr,*temp;
        int new_data;
        ptr = (struct node *)malloc(sizeof(struct node));

        if(ptr == NULL)
        {
            printf("\nOVERFLOW!!!\n");
        }
        else
        {
            printf("\nEnter the Value to be inserted:");
            scanf("%d",&new_data);
            ptr->data = new_data;
            if(head == NULL)
```

```
                {
                        ptr->next =
                        NULL; head =
                        ptr;
                }       printf("\nNODE INSERTED\n");
                else
                {

                        temp = head;
                        while(temp->next != NULL)
                        {
                                temp = temp->next;
                        }
                        temp->next = ptr;
                        ptr->next =
                        NULL;
                        printf("\nNODE INSERTED AT THE END\n");
                }
        }
}

void insert_specificpos()
{
        int i,position,new_data;
        struct node *ptr,*temp;
        ptr = (struct node *)malloc(sizeof(struct node));

        if(ptr == NULL)
        {
                printf("\nOVERFLOW!!!\n");
        }
```

```
        else
        {
                printf("\nEnter the Value to be inserted:");
                scanf("%d",&new_data);
                ptr->data = new_data;
                printf("\nEnter the position to insert the element:");
                scanf("%d",&position);
                temp = head;
                if(position ==
                1)
                {
                        ptr->next = temp;
                        head = ptr; return;
                }
                for(i=1;i<position-1;i++)
                {
                        temp = temp->next;
                }
                ptr->next = temp->next;
                temp->next = ptr;
                printf("\nNODE INSERTED AT %d POSITION \n",position);
        }
}

void delete_front()
{
        struct node *ptr;
        if(head ==
        NULL)
        {
```

```
                printf("EMPTY LIST!!!");
        }
        else
        {
                ptr = head;
                head = ptr->next;
                free(ptr);
                printf("\nNODE DELETED FROM THE BEGINING\n");
        }
}

void delete_end()
{
        struct node *ptr,*ptr1;
        if(head == NULL)
        {
                printf("EMPTY LIST!!!");
        }
        else if(head->next == NULL)
        {
                head =
                NULL;
                free(head);
        }       printf("\nONLY NODE IN THE LIST DELETED\n");
        else
        {

                ptr = head;
                while(ptr->next != NULL)
                {
```

```
                ptr1 = ptr;
                ptr = ptr->next;
            }
            ptr1->next =
            NULL; free(ptr);
            printf("\nNODE DELETED FROM THE END\n");
        }
}

void delete_specificpos()
{
        struct node *ptr,*ptr1;
        int position,i;
        printf("\nEnter the position to delete the element:");
        scanf("%d",&position);
        ptr = head;
        for(i=0;i<position;i++
        )
        {
            ptr1 = ptr;
            ptr = ptr->next;

            if(ptr == NULL)
            {
                printf("\nLess Than Required Elements in the
                List!"); return;
            }
        }
        ptr1->next = ptr->next;
        free(ptr);
```

```
        printf("\nNODE DELETED %d\n",position+1);
}

void display_list()
{
        struct node *ptr;
        ptr = head;
        if(ptr == NULL)
        {
                printf("EMPTY LIST!!!INSERT FEW ELEMENTS!!");
        }
        else
        {
                printf("\n\nLIST--
                >"); while(ptr !=
                NULL)
                {
                        printf("\t%d",ptr->data);
                        ptr = ptr->next;
                }
        }
}
int main()
{
        int choice =
        0; while(1)
        {
                printf("\n\n----------MENU--------- \n");
                printf("Choose an option from the list:");
```

```
            printf("\n[1]Insert in the Begining\n[2]Insert at the End\n[3]Insert at Specific Position\n[4]Delete from the Begining\n[5]Delete from the End\n[6]Delete at a Specific Position\n[7]Display Linked List\n[8]EXIT\n");
            printf("\nEnter your choice:");
            scanf("%d",&choice);
            switch(choice)
            {
                case 1: insert_front();
                        break;
                case 2: insert_end();
                        break;
                case 3: insert_specificpos();
                        break;
                case 4: delete_front();
                        break;
                case 5: delete_end();
                        break;
                case 6: delete_specificpos();
                        break;
                case 7: display_list();
                        break;
                case 8: exit(1);
                default:
                    printf("\nINVALID CHOICE!!!\n");

            }
        }
}
```

OUTPUT

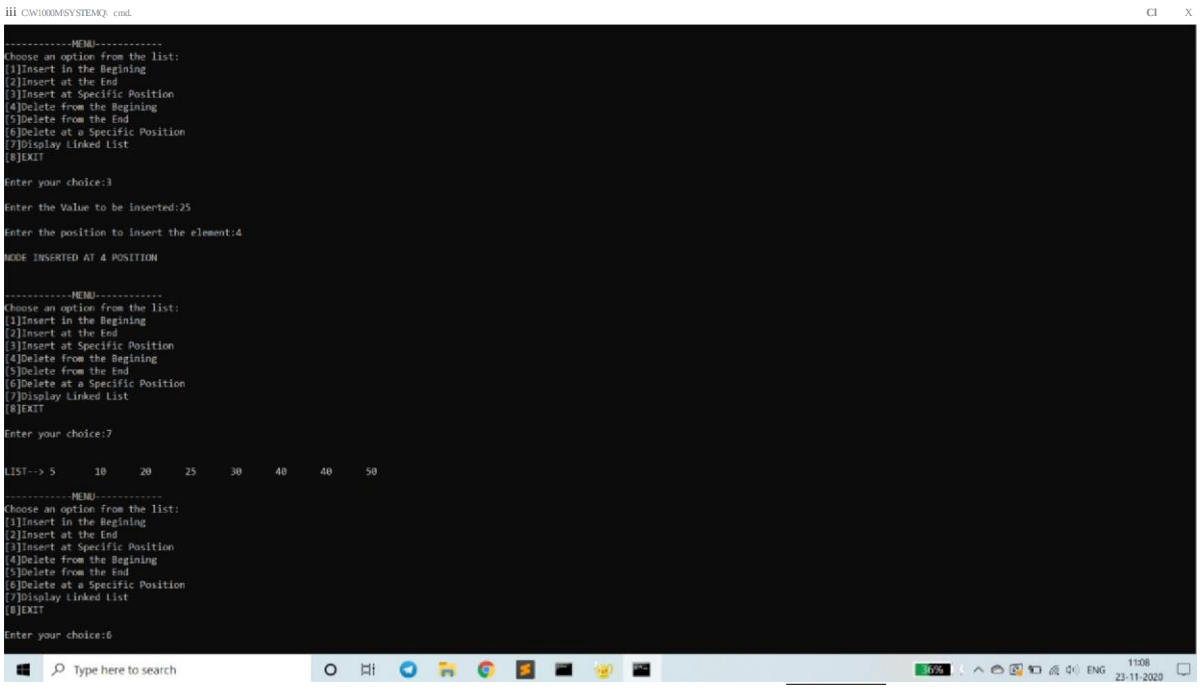
```
------------MENU------------
Choose ar option from the list:
[1]Insert in the Begiring
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT

Enter your choice:1

Enter the Value to be inserted:10

NODE INSERTED AT THE FRONT


------------MENU------------
Choose ar option from the list:
[1]insert in the Begiring
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT

Enter your choice:7


LIST--> 10

------------MENU------------
Choose ar option from the list:
[1]Insert in the Begiring
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT

Enter your choice:2

Enter the Value to be inserted:20
```

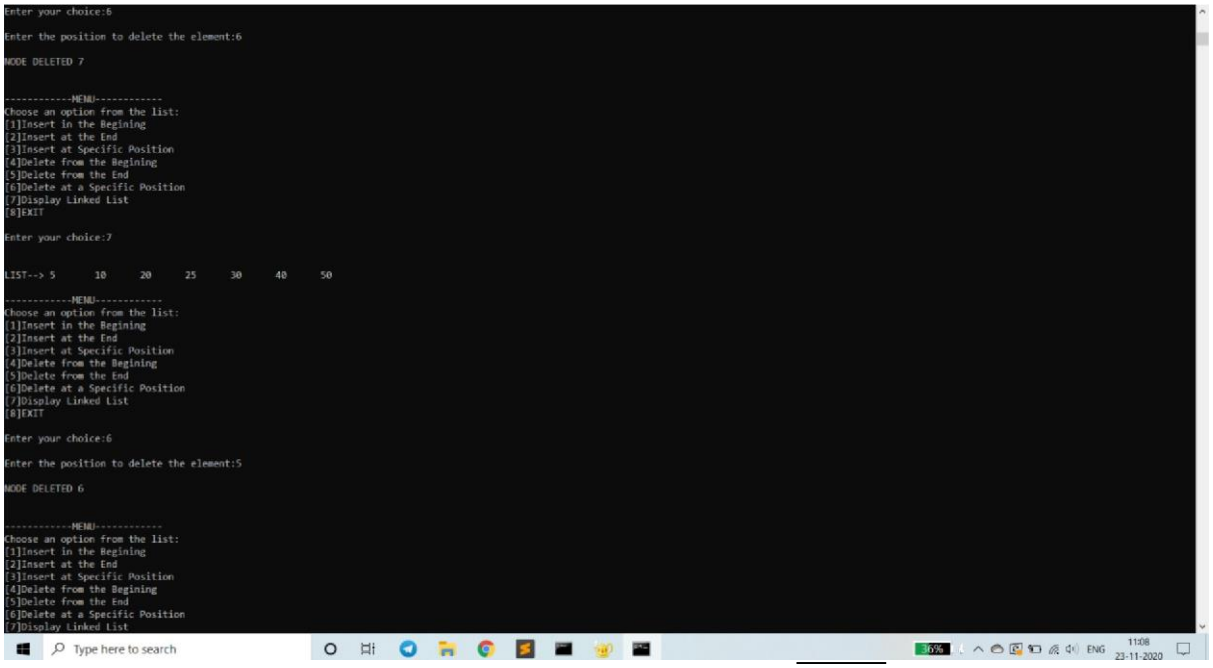
```
Enter the Value to be inserted:20

NODE INSERTED AT THE END


------------MENU------------
Choose ar option from the list:
[1]Insert in the Begiring
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT

Enter your choice:2

Enter the Value to be inserted:30

NODE INSERTED AT THE END


------------MENU------------
Choose ar option from the list:
[1]Insert in the Begiring
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT

Enter your choice:2

Enter the Value to be inserted:40

NODE INSERTED AT THE END


------------MENU------------
Choose ar option from the list:
[1]Insert in the Begiring
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
```

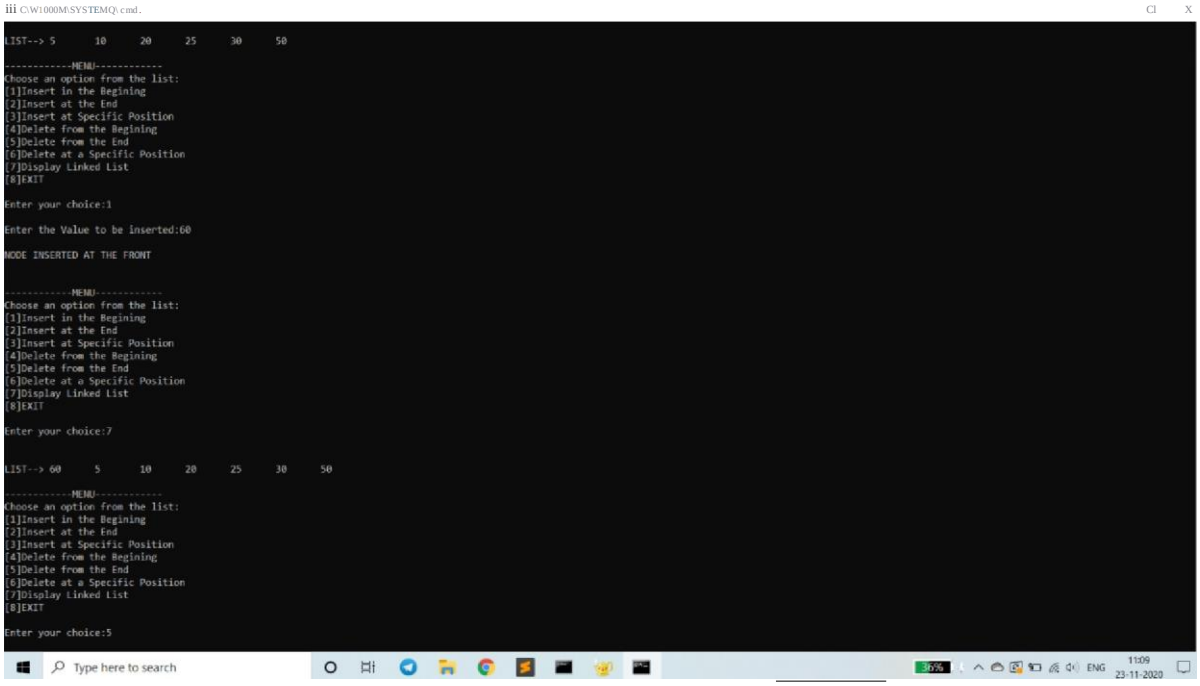
```
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT

Enter your choice:7


LIST--> 10      20      30      40      40      50

------------MENU------------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT

Enter your choice:1

Enter the Value to be inserted:5

NODE INSERTED AT THE FRONT


------------MENU------------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT

Enter your choice:7


LIST--> 5       10      20      30      40      40      50
```

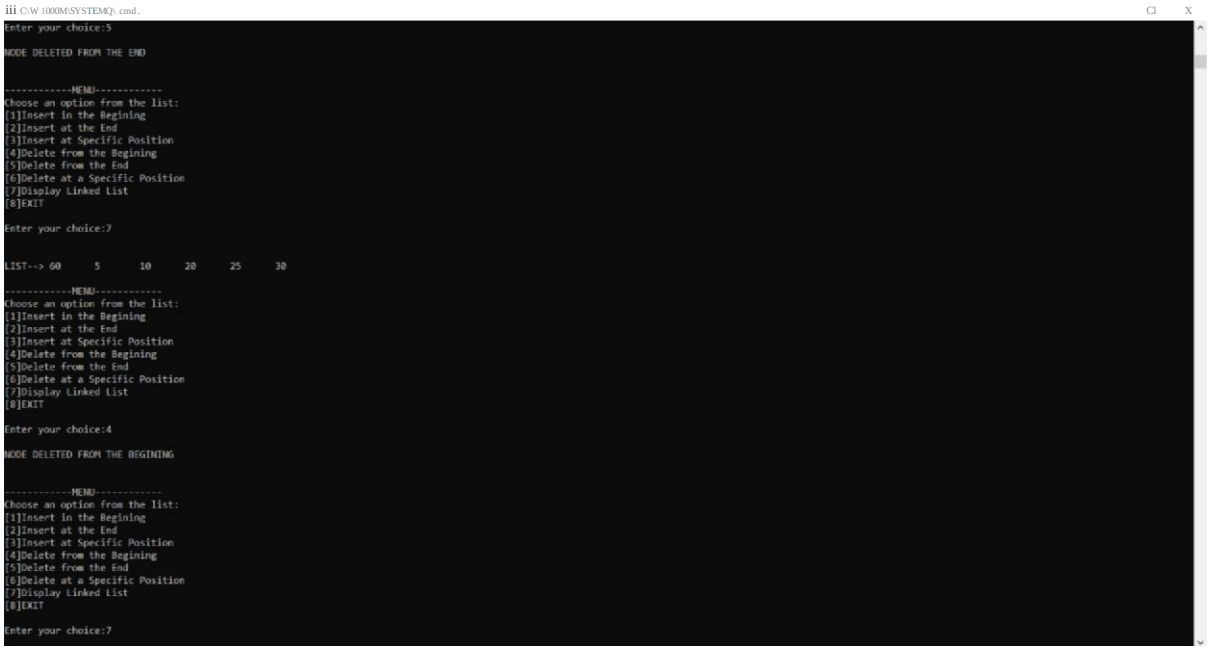
x

```
------------MENU------------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT

Enter your choice:2

Enter the Value to be inserted:40

NODE INSERTED AT THE END


------------MENU------------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT

Enter your choice:2

Enter the Value to be inserted:50

NODE INSERTED AT THE END


------------MENU------------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT

Enter your choice:7
```


C:\W1000MSYSTEM\Q. cmd
----------MENU----------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT
Enter your choice:3
Enter the Value to be inserted:25
Enter the position to insert the element:4
NODE INSERTED AT 4 POSITION
----------MENU----------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT
Enter your choice:7
LIST--> 5 10 20 25 30 40 40 50
----------MENU----------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT
Enter your choice:6
Type here to search
ENG
11:08
23-11-2020


C:\W1000MSYSTEM\32cmd
Enter your choice:6
Enter the position to delete the element:6
NODE DELETED 7
----------MENU----------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT
Enter your choice:7
LIST--> 5 10 20 25 30 40 50
----------MENU----------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT
Enter your choice:6
Enter the position to delete the element:5
NODE DELETED 6
----------MENU----------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
Type here to search
ENG
11:08
23-11-2020


iii C:\W1000M\SYSTEMQ\cmd.
LIST--> 5 10 20 25 30 50
----------MENU----------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT
Enter your choice:1
Enter the Value to be inserted:60
NODE INSERTED AT THE FRONT
----------MENU----------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT
Enter your choice:7
LIST--> 60 5 10 20 25 30 50
----------MENU----------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT
Enter your choice:5
Type here to search
ENG
11:09
23-11-2020


iii C:\W1000M\SYSTEMQ\cmd.
Enter your choice:5
NODE DELETED FROM THE END
----------MENU----------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT
Enter your choice:7
LIST--> 60 5 10 20 25 30
----------MENU----------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT
Enter your choice:4
NODE DELETED FROM THE BEGINING
----------MENU----------
Choose an option from the list:
[1]Insert in the Begining
[2]Insert at the End
[3]Insert at Specific Position
[4]Delete from the Begining
[5]Delete from the End
[6]Delete at a Specific Position
[7]Display Linked List
[8]EXIT
Enter your choice:7

LAB 7 & LAB 8

Q. WAP Implement Singly Linked List with following operations

a) Sort the linked list.

b) Reverse the linked list.

c) Concatenation of two linked lists

d) Implementation of Stacks & Queues using Linked Lists

PROGRAM

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

struct node
{
        int data;
        struct node *next;
};
struct node *head;
struct node *head2;

//stack operations
void push()
{
        struct node *ptr;
        int new_data;
        ptr = (struct node *)malloc(sizeof(struct node));
```

```
        if(ptr == NULL)
        {
                printf("\nOVERFLOW!!!");
        }
        else
        {
                printf("\nEnter the Value to be inserted:");
                scanf("%d",&new_data);
                ptr->data = new_data;
                ptr->next = head; head
                = ptr;
                printf("\nNODE INSERTED AT THE TOP OF THE STACK\n");
        }
}
void pop()
{
        struct node *ptr;
        if(head ==
        NULL)
        {
                printf("EMPTY LIST!!!");
        }
        else
        {
                ptr = head;
                head = ptr->next;
                free(ptr);
                printf("\nNODE DELETED FROM TOP OF THE STACK\n");
        }
}
```

```
//queue operations
void enqueue()
{
        struct node *ptr,*temp;
        int new_data;
        ptr = (struct node *)malloc(sizeof(struct node));

        printf("\nEnter the Value to be inserted:");
        scanf("%d",&new_data);
        ptr->data = new_data;
        if(head == NULL)
        {
                ptr->next =
                NULL; head =
                ptr;
        }       printf("\nNODE INSERTED AT REAR OF THE QUEUE\n");
        else
        {

                temp = head;
                while(temp->next != NULL)
                {
                        temp = temp->next;
                }
                temp->next = ptr;
                ptr->next =
                NULL;
                printf("\nNODE INSERTED AT REAR OF THE QUEUE\n");
        }
}
```

```
void dequeue()
{
        struct node *ptr;
        if(head ==
        NULL)
        {
                printf("EMPTY LIST!!!");
        }
        else
        {
                ptr = head;
                head = ptr->next;
                free(ptr);
                printf("\nNODE DELETED FROM FRONT OF THE QUEUE\n");
        }
}

//Display List
void
display()
{
        struct node *ptr;
        ptr = head;
        if(ptr == NULL)
        {
                printf("EMPTY LIST!!!INSERT FEW ELEMENTS!!");
        }
        else
        {
                printf("\n\nLIST--
                >"); while(ptr !=
                NULL)
```

```
                {
                        printf("\t%d",ptr->data);
                        ptr = ptr->next;
                }
        }
}

//sort Linked list in ascending order
void sort()
{
        struct node *ptr = head;
        struct node *temp =
        NULL; int i;

        if(head == NULL)
        {
                return;
        }
        else
        {
                while(ptr != NULL)
                {
                        temp = ptr->next;
                        while(temp !=
                        NULL)
                        {
                                if(ptr->data >temp->data)
                                {
                                        i = ptr->data;
                                        ptr->data = temp->data;
```

```
                                        temp->data = i;
                                }
                                temp = temp->next;
                        }
                        ptr = ptr->next;
                }
        }
}
//reverse Linked List
void reverse()
{
        struct node *prev =
        NULL; struct node *next
        = NULL; struct node *ptr
        = head; while(ptr !=
        NULL)
        {
                next = ptr->next;
                ptr->next = prev;
                prev = ptr;
                ptr = next;
        }
        head = prev;
}
//create list
struct node *create_list(struct node *head)
{
        struct node *ptr,*temp;
        int i,n,new_data;
   printf("\nEnter the number of nodes : ");
```

```
scanf("%d",&n);
        head =
        NULL; if(n
        == 0)
        {
                return head;
        }
        for(i=1;i<=n;i++)
        {
                ptr = (struct node *)malloc(sizeof(struct node));
                printf("Enter the element to be inserted : ");
                scanf("%d",&new_data);
                ptr->data = new_data;
                if(head == NULL)
                {
                        ptr->next =
                        NULL; head =
                }       ptr;
                else
                {

                        temp = head;
                        while(temp->next != NULL)
                        {
                                temp = temp->next;
                        }
                        temp->next = ptr;
                        ptr->next =
                        NULL;
                }
        }
        return head;
```

```
}
//concatenate two lists
struct node *concatenate(struct node *head, struct node *head2)
{
        struct node *ptr;
        if(head ==
        NULL)
        {
                head = head2;
                return head;
        }
        if(head2 == NULL)
        {
                return head;
        }
        ptr = head;
        while(ptr->next != NULL)
        {
                ptr = ptr->next;
        }
        ptr->next = head2;
        return head;
}
int main()
{
        int choice =
        0; while(1)
        {
                printf("\n\n**********MENU**********\n");
                printf("Choose an option from the list:");
```

```
        printf("\n-----STACK OPERATIONS        \n[1]PUSH\n[2]POP");
        printf("\n-----QUEUE OPERATIONS \n[3]ENQUEUE\n[4]DEQUEUE");
        printf("\n.....");
printf("\n[5]DISPLAY\n[6]SORT\n[7]REVERSE\n[8]CONCATENATION\n[9]E
XIT\n");
        printf("\nEnter your choice:");
        scanf("%d",&choice);
        switch(choice)
        {
                case 1: push();
                                break;
                case 2: pop();
                                break;
                case 3: enqueue();
                                break;
                case 4: dequeue();
                                break;
                case 5: display();
                                break;
                case 6: sort();
                                printf("\nSorted List::");
                                display();
                                break;
                case 7: reverse();
                                printf("\nReversed
                                List::"); display();
                                break;
                case 8: printf("\nCreate a Second list-->");
                                head2 = create_list(head2);
                                printf("\nList1:");
```

```
                        display();
                        struct node *ptr;
                        ptr = head2;
                        if(ptr == NULL)
                        {
                                printf("LIST2 IS EMPTY!!!");
                        }
                        else
                        {
                                printf("\n\nLIST2--
                                >"); while(ptr !=
                                NULL)
                                {
                                        printf("\t%d",ptr->data);
                                        ptr = ptr->next;
                                }
                        }
                        head = concatenate(head,head2);
                        printf("\n\nConcatenated List:");
                        display();
                        break;
                case 9: exit(1);
                default:
                        printf("\nINVALID CHOICE!!!\n");

                }
        }
}
```

OUTPUT

```
****************MENU***************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
-------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:1

Enter the Value to be inserted:10

NODE INSERTED AT THE TOP OF THE STACK


****************MENU***************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
```

```
[3]ENQUEUE
[4]DEQUEUE
-------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:3

Enter the Value to be inserted:20

NODE INSERTED AT REAR OF THE QUEUE


****************MENU***************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
-------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT
```

```
[8]CONCATENATION
[9]EXIT

Enter your choice:2

NODE DELETED FROM TOP OF THE STACK


****************MENU***************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
-------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:5


LIST--> 20

****************MENU***************
Choose an option from the list:
```

```
****************MENU***************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
[4]DEQUEUE
-------------------------
[5]DISPLAY
[6]SORT
[7]REVERSE
[8]CONCATENATION
[9]EXIT

Enter your choice:1

Enter the Value to be inserted:10

NODE INSERTED AT THE TOP OF THE STACK


****************MENU***************
Choose an option from the list:
-----STACK OPERATIONS-----
[1]PUSH
[2]POP
-----QUEUE OPERATIONS-----
[3]ENQUEUE
```

LAB 9

Q. WAP Implement doubly link list with primitive operations

a) Create a doubly linked list.

b) Insert a new node to the left of the node.

c) Delete the node based on a specific value

d) Display the contents of the list

PROGRAM

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

struct node
{
        struct node *prev;
        int data;
        struct node *next;
};
struct node *head;
struct node *last;

//Create DLL
void create_list()
{
        struct node *ptr;
        int i,n,new_data;
        printf("\nEnter the number of nodes:");
        scanf("%d",&n);
```

```
        if(n>=1)
        {
                head = (struct node *)malloc(sizeof(struct node));
                if(head != NULL)
                {
                        printf("\nEnter the value to be inserted for Node 1 :\t");
                        scanf("%d",&new_data);

                        head->data = new_data;
                        head->prev = NULL;
                        head->next = NULL;

                        last = head;

                        for(i=2;i<=n;i++)
                        {
                                ptr = (struct node *)malloc(sizeof(struct node));

                                if(ptr != NULL)
                                {
                                        printf("Enter the value to be inserted for Node %d
:\t",i);
                                        scanf("%d",&new_data);

                                        ptr->data = new_data;
                                        ptr->prev = last;
                                        ptr->next = NULL;

                                        last->next = ptr;
```

```
                                last = ptr;
                        }
                }
                printf("\n\nLinked List Created!!");
        }
    }
    else
    {
        printf("\n\nInvalid!!Enter valid number of nodes!!");
    }
}

//display
void display_list()
{
    struct node *ptr = head;
    if(ptr == NULL)
    {
        printf("\nList is Empty!!");
    }
    else
    {
        printf("\n\nLIST--
        >"); while(ptr !=
        NULL)
        {
            printf("\t%d",ptr->data);
            ptr = ptr->next;
        }
    }
```

```
}

//Insert at the left of a given node
void insert_left()
{
	int i,pos,new_data;
	struct node *ptr,*temp;
	ptr = (struct node *)malloc(sizeof(struct node));

	printf("\nEnter the Node to insert the value: ");
	scanf("%d",&pos);
	printf("\nEnter the value to be inserted: ");
	scanf("%d",&new_data);

	ptr->data = new_data;

	if(head == NULL)
	{
		printf("\nList is empty!!");
	}
	else
	{
		temp = head;
		i=1;
		while(i<pos-1 && temp!=NULL)
		{
			temp = temp->next;
			i++;
		}
```

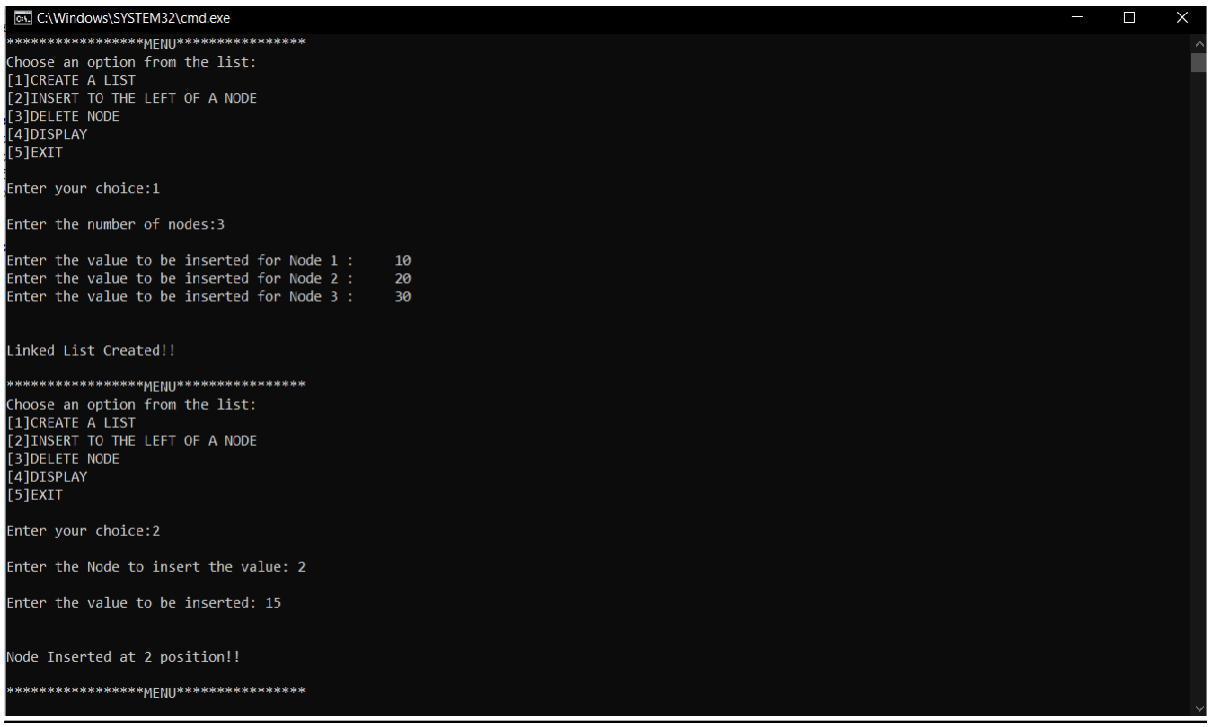
```
if(pos == 1)
{
        ptr->next = head;
        ptr->prev =
        NULL;   head-
        >prev = ptr; head
        = ptr;
        printf("\n\nNode Inserted at %d position!!",pos);
}
else if(temp == last)
{
        ptr->next =
        NULL; ptr->prev
        = last; last->next
        = ptr; last = ptr;
        printf("\n\nNode Inserted at %d position!!",pos);
}
else if(temp != NULL)
{
        ptr->next = temp->next;
        ptr->prev = temp;
        if(temp->next !=
        NULL)
        {
                temp->next->prev = ptr;
        }
        temp->next = ptr;
        printf("\n\nNode Inserted at %d position!!",pos);
}
else
{
```

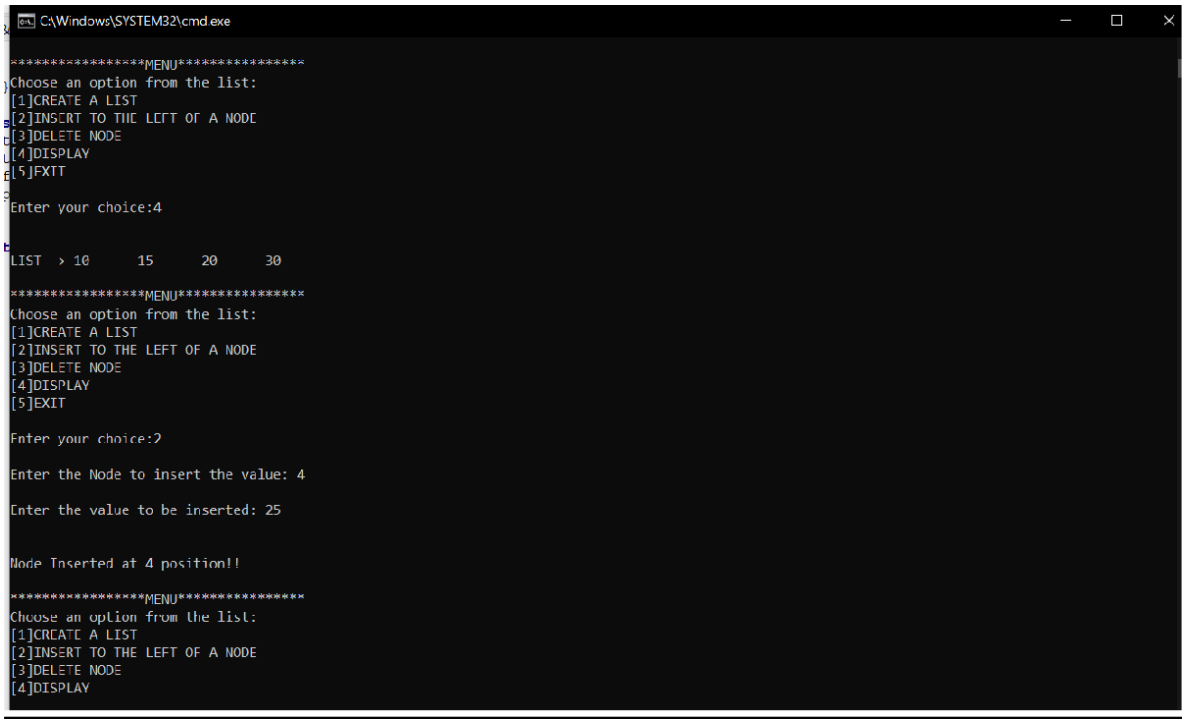
```
                printf("\n\nInvalid Position!!");
            }
        }
}

//Delete node by Value
void delete()
{
 struct node* temp = head;
 struct node* ptr = (struct node*) malloc(sizeof(struct node));
 int val;
 printf("\nEnter the Value to be deleted: ");
 scanf("%d",&val);

 if(temp->next == NULL)
 {
  head =
  NULL;
  free(temp);
  printf("\n\nValue %d, deleted \n",val);
  return;
 }

 if(temp!=NULL && temp->data == val)
 {
  head = temp->next;
  head->prev =
  NULL; free(temp);
  printf("\n\nValue %d, deleted ",val);
        return;
```

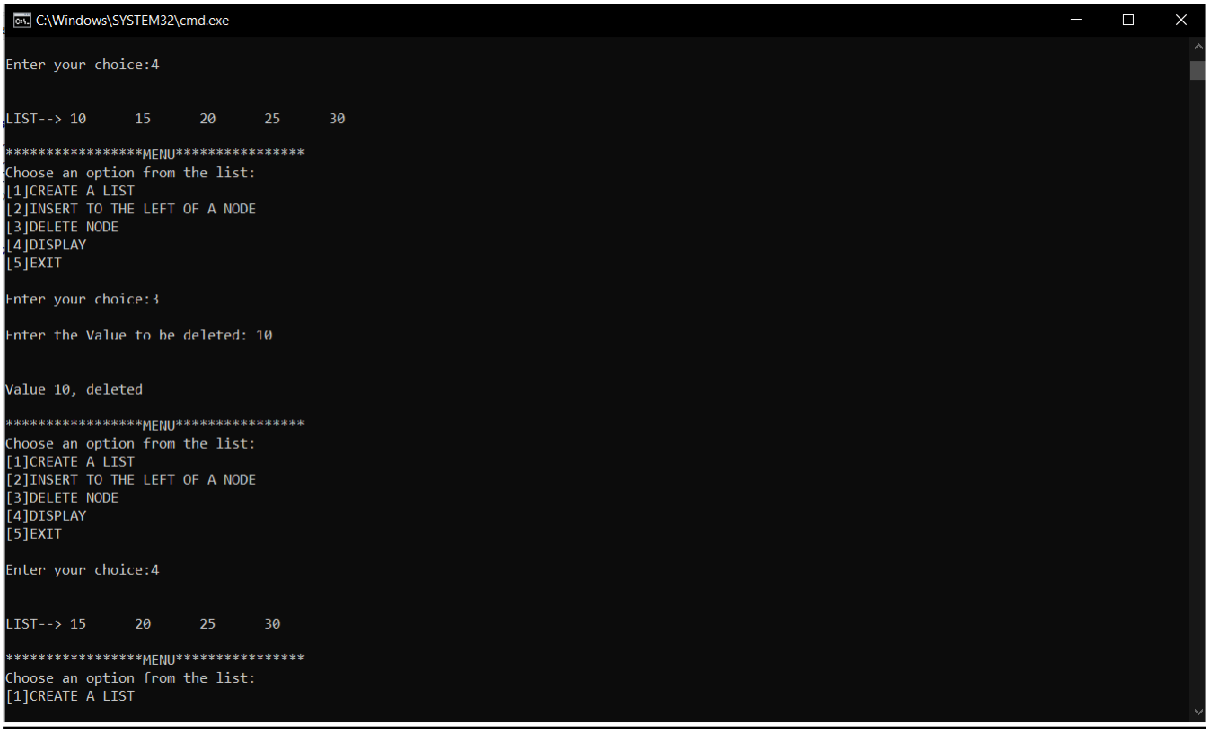
```
 }
 while(temp!=NULL && temp->data != val)
 {
  ptr = temp;
  temp = temp->next;
 }
 if(temp==NULL)
 {
  printf("\n\nValue not found");
  return;
 }
 ptr->next = temp->next;

 if(temp->next == NULL)
 {
  printf("\n\nValue %d, deleted \n",val);
  free(temp);
  return;
 }

 struct node* temp2 = (struct node*) malloc(sizeof(struct node));
 temp2 = temp->next;
 temp2->prev = ptr;
 free(temp);
 printf("Value %d, deleted \n",val);
}

int main()
{
```

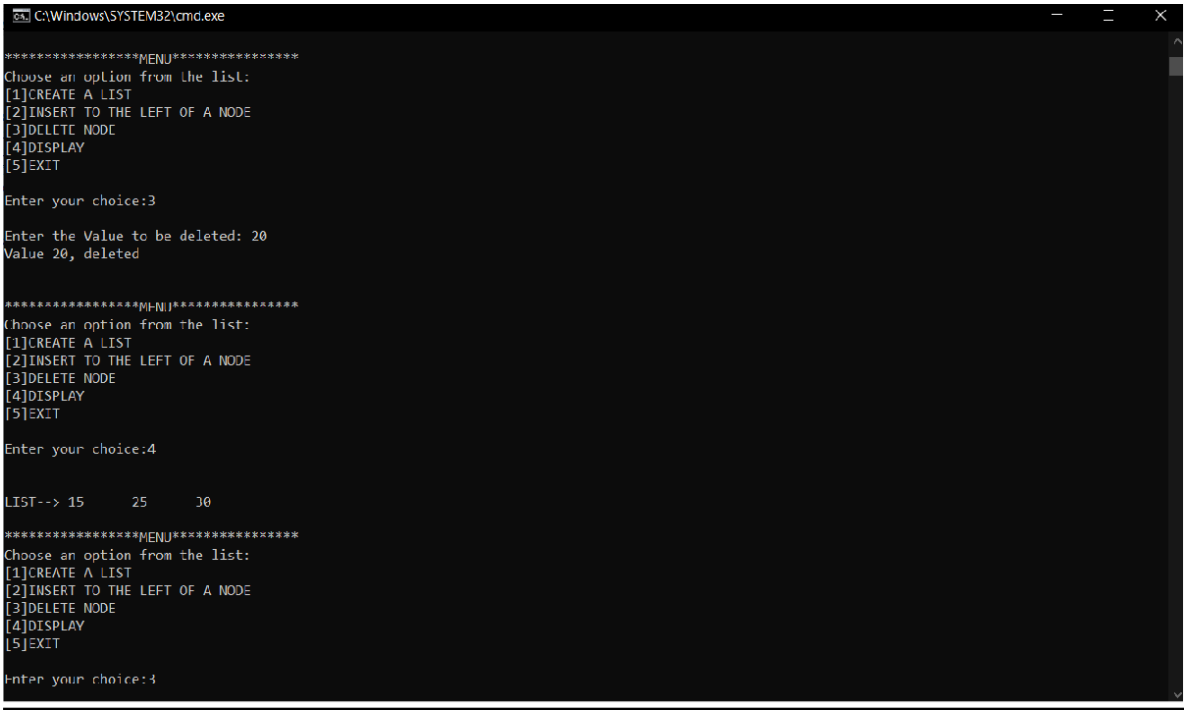
```
	int choice =
	0; while(1)
	{
		printf("\n\n*****************MENU****************\n");
		printf("Choose an option from the list:");
		printf("\n[1]CREATE A LIST\n[2]INSERT TO THE LEFT OF A
NODE\n[3]DELETE NODE \n[4]DISPLAY\n[5]EXIT\n");
		printf("\nEnter your choice:");
		scanf("%d",&choice);
		switch(choice)
		{
			case 1: create_list();
				break;
			case 2: insert_left();
				break;
			case 3: delete();
				break;
			case 4: display_list();
				break;
			case 5: exit(1);
			default:
				printf("\nINVALID CHOICE!!!\n");

		}
	}
}
```

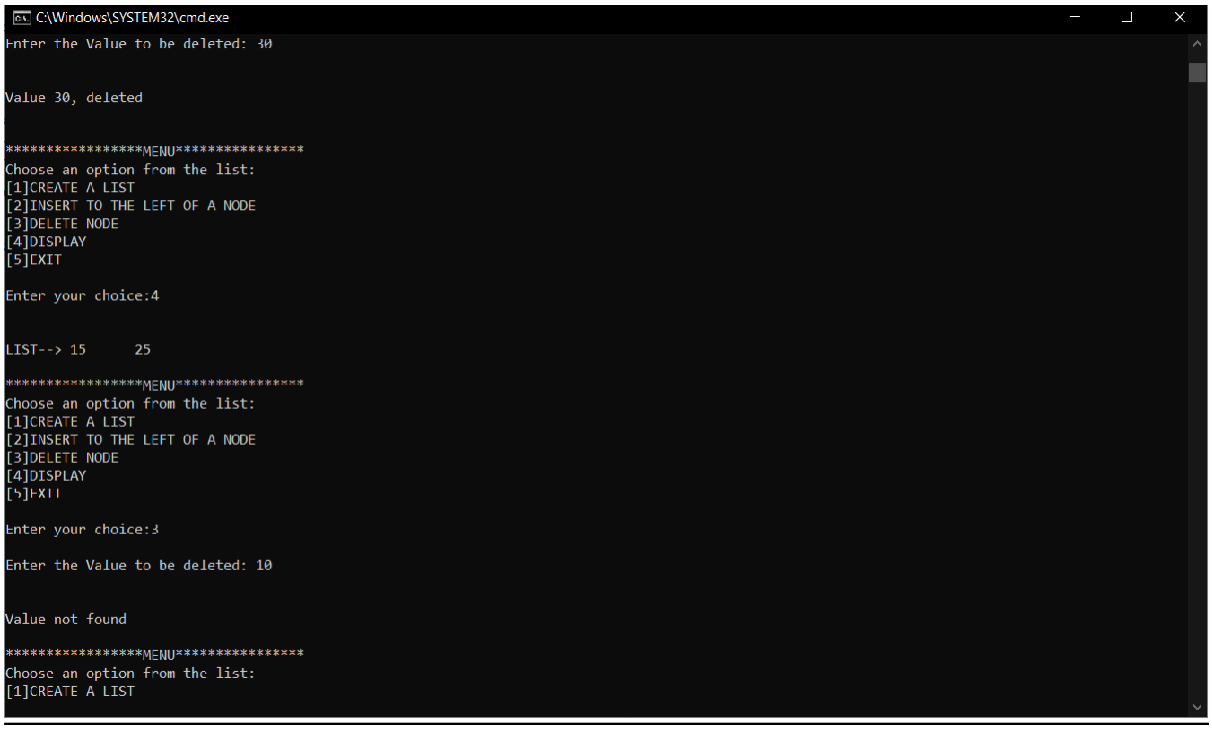
OUTPUT


C:\Windows\SYSTEM32\cmd.exe
**********MENU**********
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT
Enter your choice:1
Enter the number of nodes:3
Enter the value to be inserted for Node 1 : 10
Enter the value to be inserted for Node 2 : 20
Enter the value to be inserted for Node 3 : 30
Linked List Created!!
**********MENU**********
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT
Enter your choice:2
Enter the Node to insert the value: 2
Enter the value to be inserted: 15
Node Inserted at 2 position!!
**********MENU**********


C:\Windows\SYSTEM32\cmd.exe
**********MENU**********
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT
Enter your choice:4
LIST > 10 15 20 30
**********MENU**********
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT
Enter your choice:2
Enter the Node to insert the value: 4
Enter the value to be inserted: 25
Node Inserted at 4 position!!
**********MENU**********
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY


C:\Windows\SYSTEM32\cmd.exe
Enter your choice:4
LIST--> 10 15 20 25 30
**********MENU**********
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT
Enter your choice:3
Enter the Value to be deleted: 10
Value 10, deleted
**********MENU**********
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT
Enter your choice:4
LIST--> 15 20 25 30
**********MENU**********
Choose an option from the list:
[1]CREATE A LIST


C:\Windows\SYSTEM32\cmd.exe
**********MENU**********
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT
Enter your choice:3
Enter the Value to be deleted: 20
Value 20, deleted
**********MENU**********
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT
Enter your choice:4
LIST--> 15 25 30
**********MENU**********
Choose an option from the list:
[1]CREATE A LIST
[2]INSERT TO THE LEFT OF A NODE
[3]DELETE NODE
[4]DISPLAY
[5]EXIT
Enter your choice:3



LAB 10

Q. Write a program

a) To construct a binary Search tree.

b) To traverse the tree using all the methods i.e., in-order, preorder and post order

c) To display the elements in the tree.

PROGRAM

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

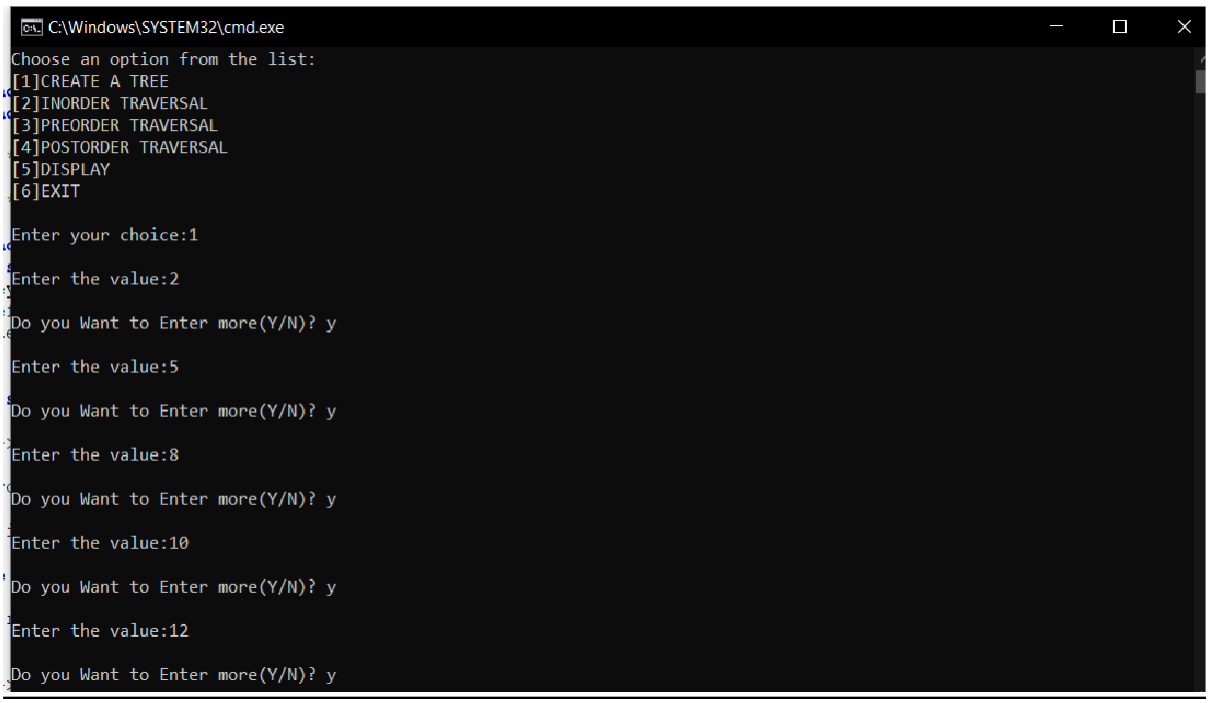
```
struct node
{
        int key;
        struct node *left;
        struct node *right;
};
struct node *root;

struct node *create(int data)
{
        struct node *temp;
        temp = (struct node*)malloc(sizeof(struct node));
        temp->key = data;
        temp->left = temp->right =
        NULL; return temp;
}

void insert(struct node *root,struct node *temp)
{
        if(temp->key < root->key)
        {
                if(root->left != NULL)
                {
                        insert(root->left,temp);
                }
                else
                {
                        root->left = temp;
                }
```

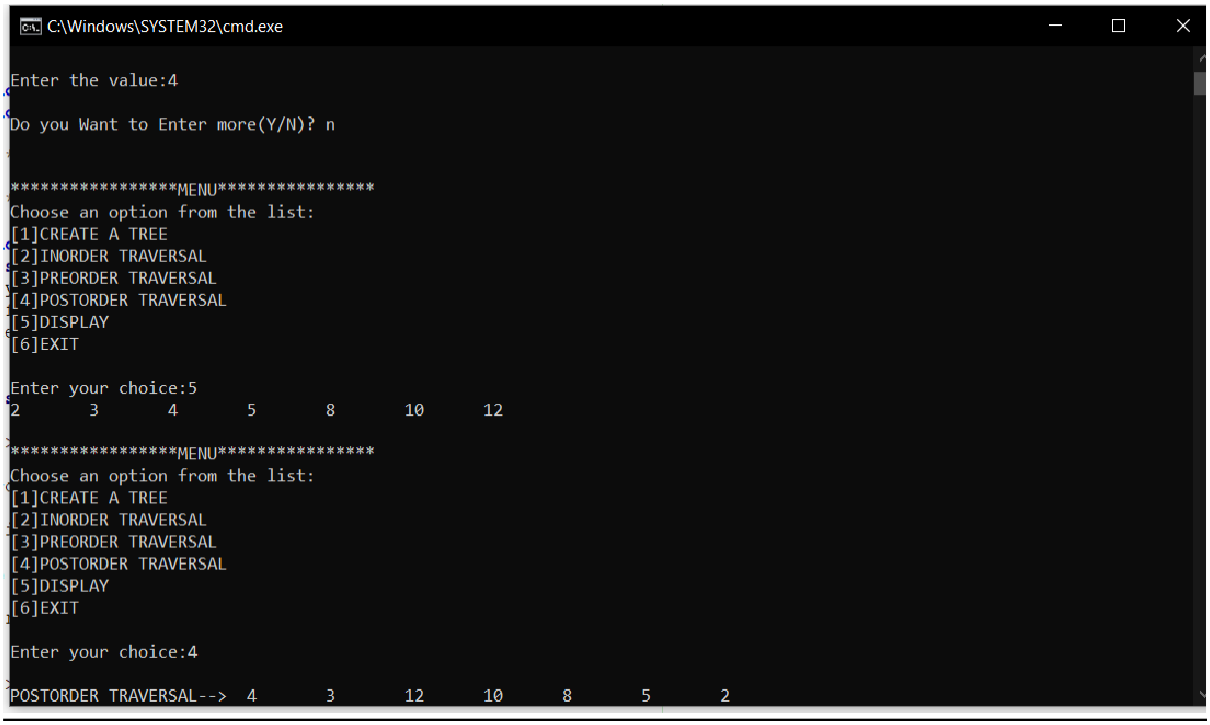
```
        }
        if(temp->key > root->key)
        {
                if(root->right != NULL)
                {
                        insert(root->right,temp);
                }
                else
                {
                        root->right = temp;
                }
        }
}

void display(struct node *root)
{
        if(root != NULL)
        {
                display(root->left);
                printf("%d\t",root->key);
                display(root->right);
        }
}

void inorder(struct node *root)
{
        if(root != NULL)
        {
                inorder(root->left);
```

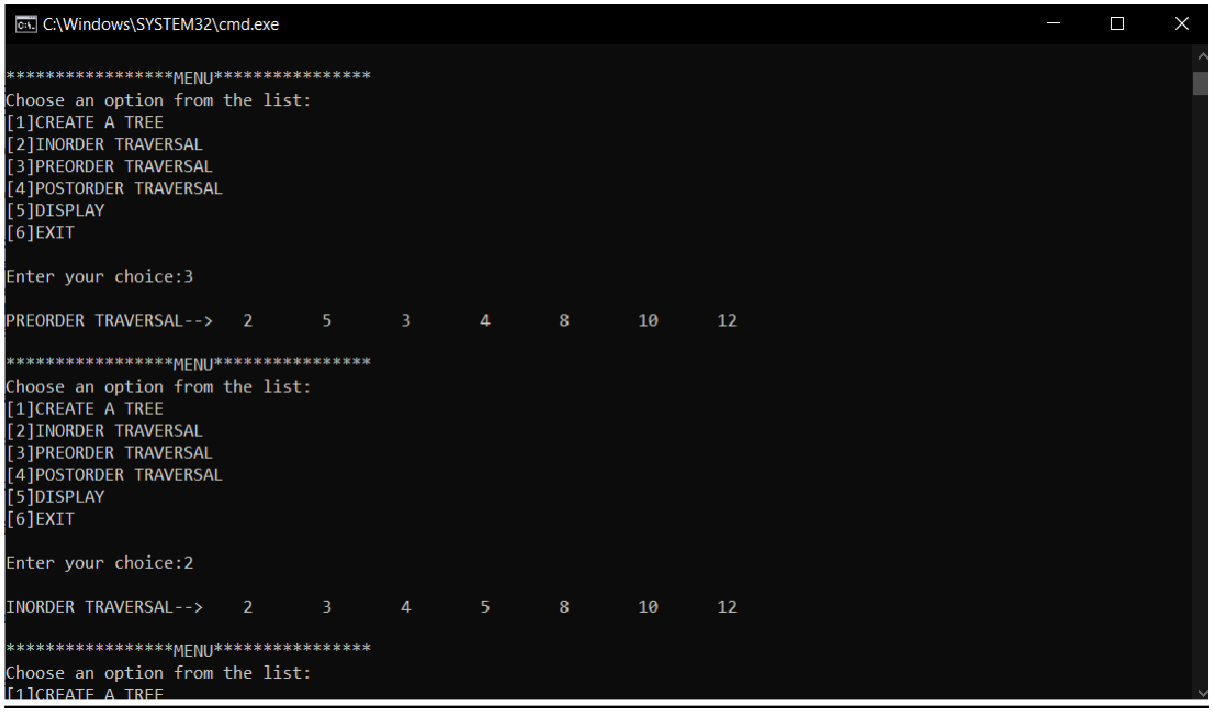
```
                printf("%d\t",root->key);
                inorder(root->right);
        }
}
void preorder(struct node *root)
{
        if(root != NULL)
        {
                printf("%d\t",root->key);
                preorder(root->left);
                preorder(root->right);
        }
}
void postorder(struct node *root)
{
        if(root != NULL)
        {
                postorder(root->left);
                postorder(root->right);
                printf("%d\t",root->key);
        }
}
int main()
{
        char ch;
        struct node *temp;
        root = NULL;
        int choice =
        0; int data;
```

```
        while(1)
        {
                printf("\n\n*****************MENU***************\n");
                printf("Choose an option from the list:");
                printf("\n[1]CREATE A TREE\n[2]INORDER
TRAVERSAL\n[3]PREORDER TRAVERSAL\n[4]POSTORDER
TRAVERSAL\n[5]DISPLAY\n[6]EXIT\n");
                printf("\nEnter your choice:");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: do{
                                                printf("\nEnter the value:");
                                                scanf("%d",&data);
                                                temp = create(data);
                                                if(root == NULL)
                                                {
                                                        root = temp;
                                                }
                                                else
                                                {
                                                        insert(root,temp);
                                                }
                                                printf("\nDo you Want to Enter more(Y/N)? ");
                                                getchar();
                                                scanf("%c",&ch);
                                        }while(ch=='y'||ch=='Y')
                                        ; break;
```

```
                case 2: printf("\nINORDER TRAVERSAL--
                        >\t"); inorder(root);
                        break;

                case 3: printf("\nPREORDER TRAVERSAL--
                        >\t"); preorder(root);
                        break;

              case 4: printf("\nPOSTORDER TRAVERSAL-->\t");
                        postorder(root);
                        break;

                case 5: display(root);
                         break;

                case 6: exit(1);
                default:
                      printf("\nINVALID CHOICE!!!\n");

            }
        }
        return 0;
}
```


C:\Windows\SYSTEM32\cmd.exe
Choose an option from the list:
[1]CREATE A TREE
[2]INORDER TRAVERSAL
[3]PREORDER TRAVERSAL
[4]POSTORDER TRAVERSAL
[5]DISPLAY
[6]EXIT
Enter your choice:1
Enter the value:2
Do you Want to Enter more(Y/N)? y
Enter the value:5
Do you Want to Enter more(Y/N)? y
Enter the value:8
Do you Want to Enter more(Y/N)? y
Enter the value:10
Do you Want to Enter more(Y/N)? y
Enter the value:12
Do you Want to Enter more(Y/N)? y


C:\Windows\SYSTEM32\cmd.exe
Enter the value:4
Do you Want to Enter more(Y/N)? n
**********MENU**********
Choose an option from the list:
[1]CREATE A TREE
[2]INORDER TRAVERSAL
[3]PREORDER TRAVERSAL
[4]POSTORDER TRAVERSAL
[5]DISPLAY
[6]EXIT
Enter your choice:5
2 3 4 5 8 10 12
**********MENU**********
Choose an option from the list:
[1]CREATE A TREE
[2]INORDER TRAVERSAL
[3]PREORDER TRAVERSAL
[4]POSTORDER TRAVERSAL
[5]DISPLAY
[6]EXIT
Enter your choice:4
POSTORDER TRAVERSAL--> 4 3 12 10 8 5 2


C:\Windows\SYSTEM32\cmd.exe
**********MENU**********
Choose an option from the list:
[1]CREATE A TREE
[2]INORDER TRAVERSAL
[3]PREORDER TRAVERSAL
[4]POSTORDER TRAVERSAL
[5]DISPLAY
[6]EXIT
Enter your choice:3
PREORDER TRAVERSAL--> 2 5 3 4 8 10 12
**********MENU**********
Choose an option from the list:
[1]CREATE A TREE
[2]INORDER TRAVERSAL
[3]PREORDER TRAVERSAL
[4]POSTORDER TRAVERSAL
[5]DISPLAY
[6]EXIT
Enter your choice:2
INORDER TRAVERSAL--> 2 3 4 5 8 10 12
**********MENU**********
Choose an option from the list:
[1]CREATE A TREE