VADODARA INSTITUTE OF ENGINEERING

KOTAMBI

Lab Manual

Data Structures (DS)

(Subject Code: 3130702)

(III Semester CE/IT)

Prepared By:

CE/IT Department

Practical List

1	Introduction to pointers. Call by Value and Call by reference.
2	Introduction to Dynamic Memory Allocation. DMA functions malloc(), calloc(), free() etc.
3	Implement a program for stack that performs following operation using array. a) PUSH b) POP
4	Implement a program for stack that performs following operation using array. PEEP b) CHANGE c) DISPLAY
5	Implement a program to convert infix notation to postfix notation using stack.
6	Write a program to implement simple queue using arrays that performs following operations (a) INSERT (b) DELETE (c) DISPLAY
7	Write a program to implement Circular Queue using arrays that performs following Operations. (a) INSERT (b) DELETE (c) DISPLAY
8	Write a menu driven program to implement following operation on the singly linked list. a) Insert a node at the front of the linked list.
9	Write a menu driven program to implement following operation on the singly linked list. a) Insert a node at the end of the linked list.
10	Write a menu driven program to implement following operation on the singly linked list. a) Delete a first node of the linked list.
11	Write a menu driven program to implement following operation on the singly linked list. a) Delete a node before specified position.
12	Write a menu driven program to implement following operation on the singly linked list. a) Delete a node after specified position
13	Write a program to implement stack using linked list.
14	Write a program to implement queue using linked list.
15	Write a program to implement following operations on the doubly linked list. a) Insert a node at the front of the linked list.
16	Write a program to implement following operations on the doubly linked list. a) Insert a node at the end of the linked list.
17	Write a program to implement following operations on the doubly linked list. a) Delete a last node of the linked list.
18	Write a program to implement following operations on the doubly linked list. a) Delete a node after specified position.
19	Write a program to implement following operations on the circular linked list. a) Insert a node at the end of the linked list.
20	Write a program to implement following operations on the circular linked list. a) Insert a node at specified position.
21	Write a program to implement following operations on the circular linked list. a) Delete a first node.

22	Write a program to implement following operations on the circular linked list. a) Delete the last node.
23	Implement recursive or non-recursive tree traversing methods of Inorder traversal.
24	Implement recursive or non-recursive tree traversing methods of Preorder traversal.
25	Implement recursive or non-recursive tree traversing methods of Postorder traversal.
26	Write a program to implement Merge Sort
27	Write a program to implement Bubble Sort
28	Write a program to implement Selection Sort

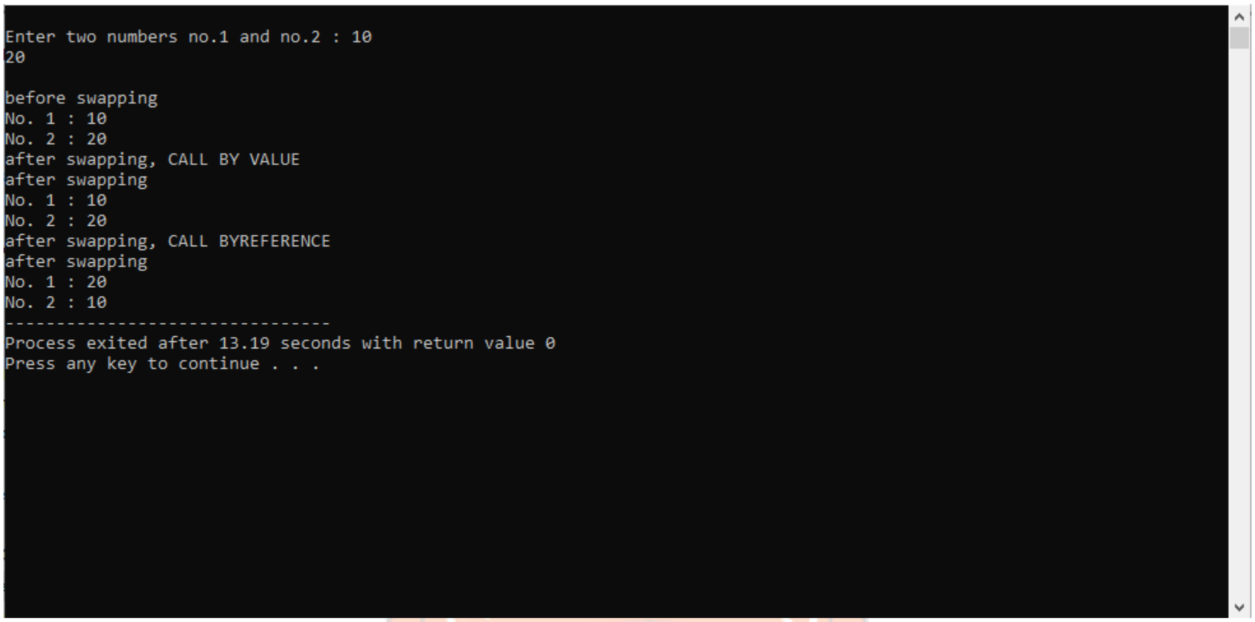
Practical – 1

AIM: Introduction to pointers. Call by Value and Call by reference.

PROGRAM:

```
include<stdio.h>
#include<conio.h>
void swap(int*num1,int*num2)
{
        int temp ;
        temp =*num1 ;
        *num1 =*num2 ;
        *num2 = temp ;
}
void swapp(int num1,int num2)
{
        int temp;
        temp=num1;
        num1=num2;
        num2=temp;
}
void main()
{
        int num1,num2;
        clrscr();
        printf("\nEnter two numbers no.1 and no.2 : ");
        scanf("%d %d",&num1,&num2);

        printf("\nbefore swapping");
        printf("\nNo. 1 : %d",num1);
        printf("\nNo. 2 : %d",num2);
        printf(" \nafter swapping, CALL BY VALUE");
        swapp(num1,num2);
        printf("\nafter swapping");
        printf("\nNo. 1 : %d",num1);
        printf("\nNo. 2 : %d",num2);
        printf(" \nafter swapping, CALL BYREFERENCE");
        swap(&num1,&num2);
        printf("\nafter swapping");
        printf("\nNo. 1 : %d",num1);
        printf("\nNo. 2 : %d",num2);
        getch();
}
```

OUTPUT:



```
Enter two numbers no.1 and no.2 : 10
20

before swapping
No. 1 : 10
No. 2 : 20
after swapping, CALL BY VALUE
after swapping
No. 1 : 10
No. 2 : 20
after swapping, CALL BYREFERENCE
after swapping
No. 1 : 20
No. 2 : 10
--------------------------------
Process exited after 13.19 seconds with return value 0
Press any key to continue . . .
```

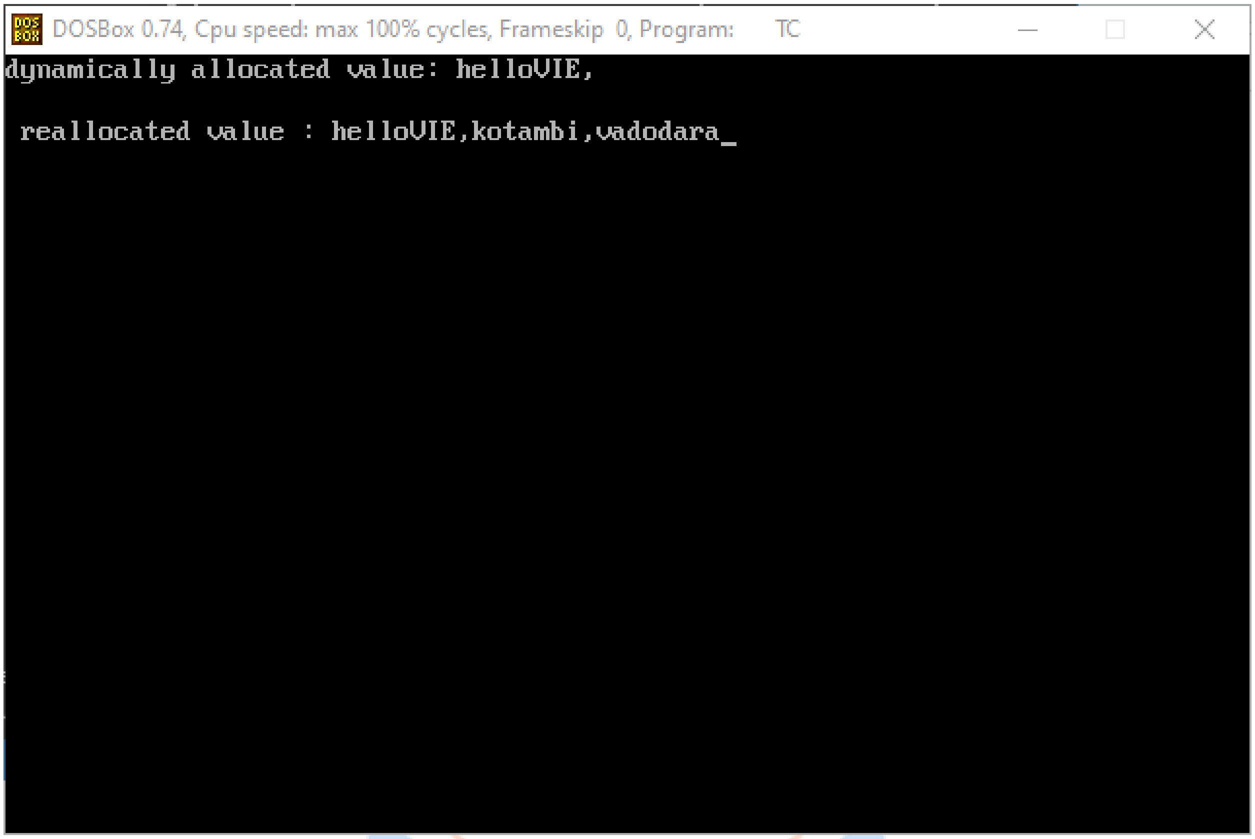
Practical – 2

AIM: Introduction to Dynamic Memory Allocation. DMA functions malloc(), calloc(), free() etc.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
void main()
{
        Char *ma;
        Clrscr();
        //allocating memory space.
        ma = malloc(sizeof(char));
        ma=”hello”;
        strcat(ma,”VIE,”);
        printf(“dynamically allocated value: %s\n”ma);
        // reallocating memory space.
        ma = reallocate(ma,100*sizeof(char));
        strcat(ma,“kotambi,vadodara”);
        printf(“\n reallocated value : %s”,ma);
        free();
        getch();
}
```

OUTPUT:



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
dynamically allocated value: helloVIE,

 reallocated value : helloVIE,kotambi,vadodara_
```

Practical – 3

AIM: To perform PUSH and POP operations on Stack.

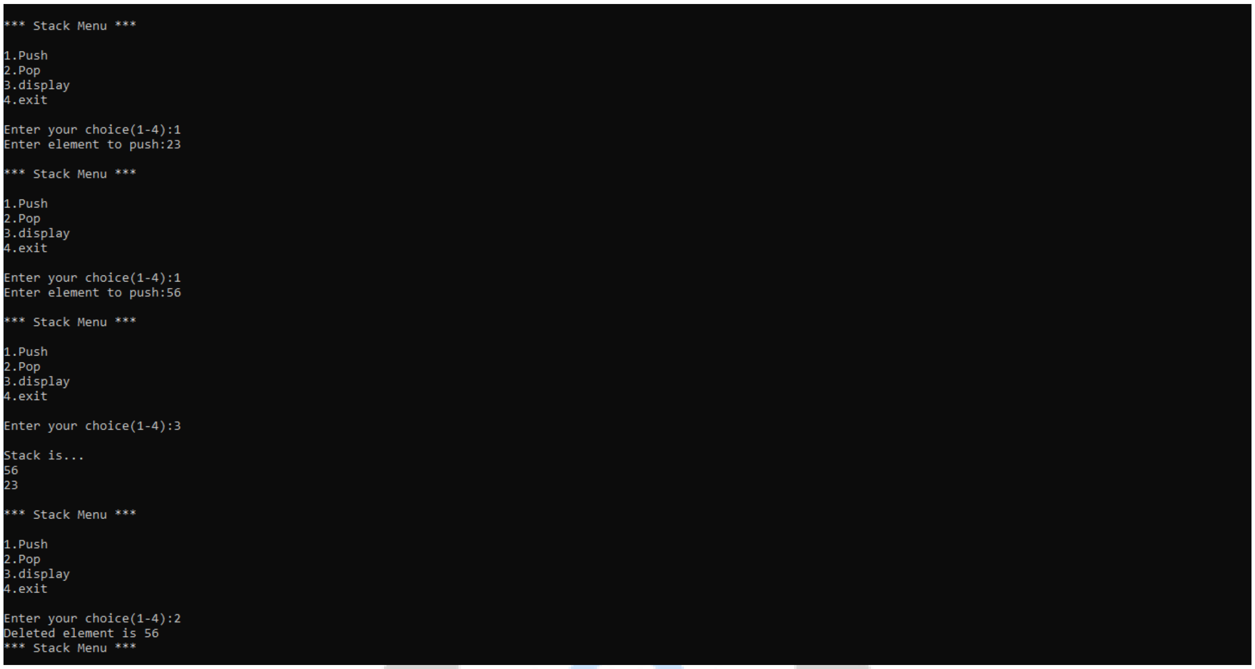
PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
//stdlib for exit function
#include<conio.h>
#define MAX 10
int top=-1,stack[MAX];
void push();
void pop();
void display();

void main()
{
 int ch;

            while(1)
            {
                        printf("\n*** Stack Menu ***"); printf("\n\n1.Push\n2.Pop\n3.display\n4.exit");
                        printf("\n\nEnter your choice(1-4):"); scanf("%d",&ch);

                        switch(ch)
                        {
                                    case 1: push();
                                    break;
                                    case 2: pop();
                                    break
                                    case 3: display();
                                    break;
                                    case 4: exit(0);
                                    break;
                                    default: printf("\nWrong Choice!!");
                        }
            }
}
```

```
void push()
{
    int val;
    if(top==MAX-1)
    {
        printf("\nStack is full!!");
    }
    else
    {
        printf("\nEnter element to push:");
        scanf("%d",&val);
        top=top+1; //top=0
        stack[top]=val;//stack[0]=10
    }
}

void pop()
{
    if(top==-1)
    {
        printf("\nStack is empty!!");
    }
    else
    {
        printf("\nDeleted element is %d",stack[top]); top=top-1;
    }
}
void display()
{
    int i;

    if(top==-1)
    {
        printf("\nStack is empty!!");
    }
```

```
        else
        {
                printf("\nStack is...\n");
                for(i=top;i>=0;--i)
                printf("%d\n",stack[i]);
        }
}
```

OUTPUT:



```
*** Stack Menu ***

1.Push
2.Pop
3.display
4.exit

Enter your choice(1-4):1
Enter element to push:23

*** Stack Menu ***

1.Push
2.Pop
3.display
4.exit

Enter your choice(1-4):1
Enter element to push:56

*** Stack Menu ***

1.Push
2.Pop
3.display
4.exit

Enter your choice(1-4):3

Stack is...
56
23

*** Stack Menu ***

1.Push
2.Pop
3.display
4.exit

Enter your choice(1-4):2
Deleted element is 56
*** Stack Menu ***
```

Practical – 4

AIM: To perform PEEP and CHANGE operations on Stack

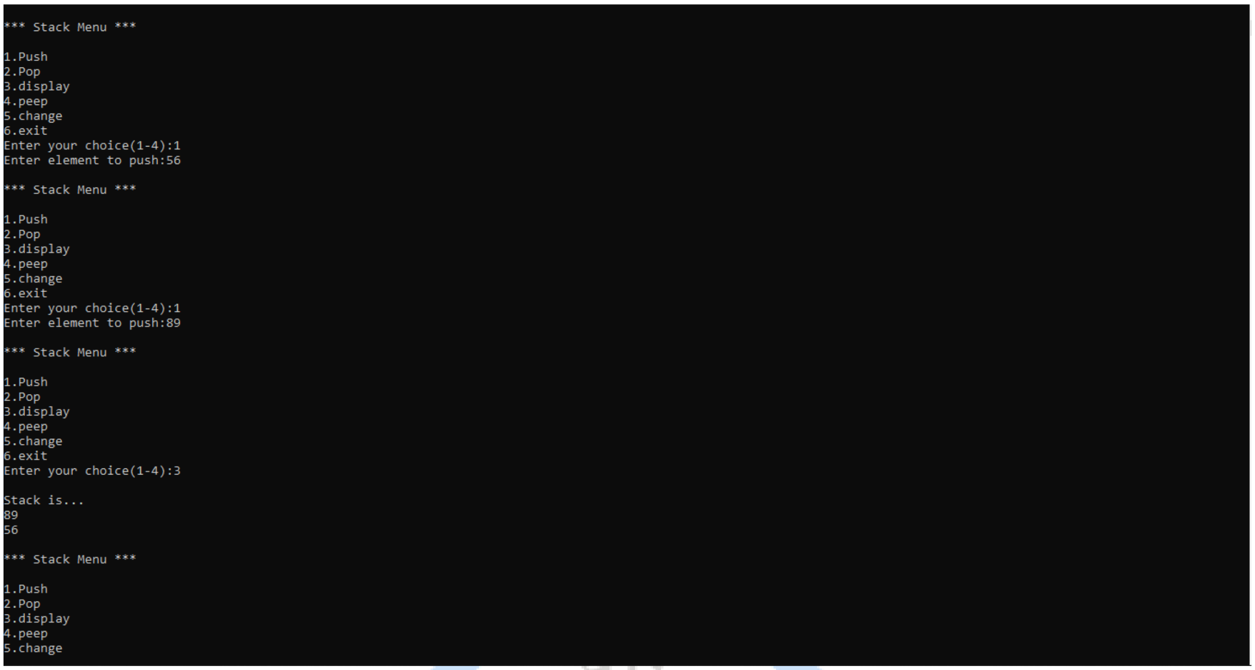
PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define MAX 5
int top=-1,stack[MAX],temp[MAX], i=-1;
void push();
void pop();
void peep();
void change();
void display();

void push()
{
int val;

if(top==MAX-1)
{
printf("\nStack is full!!");
}
else
{
printf("\nEnter element to push:");
scanf("%d",&val);
top=top+1; stack[top]=val;
}
}

void pop()
{
if(top==-1)
{
printf("\nStack is empty!!");
}
else
{
```

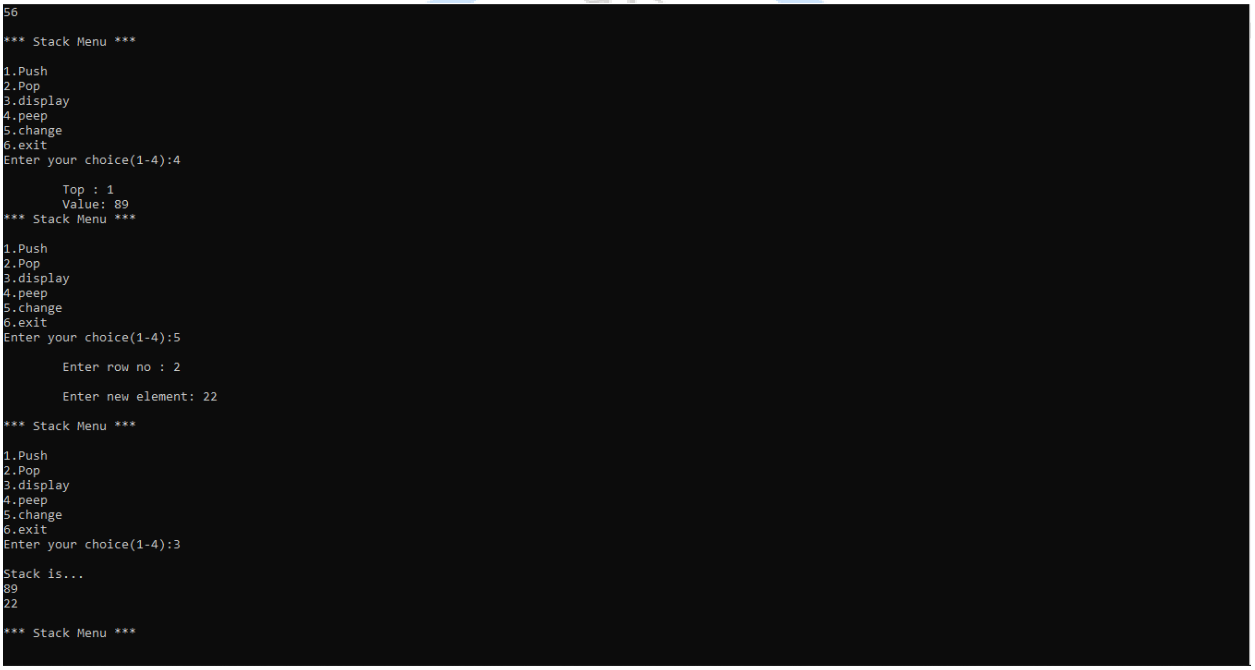
```
printf("\nDeleted element is %d",stack[top]); top=top-1;
}
}
void display()
{
int i;

if(top==-1)
{
printf("\nStack is empty!!");
}
else
{
printf("\nStack is...\n");
for(i=top;i>=0;--i)
printf("%d\n",stack[i]);
}
}
// Peep operation....
void peep(){
   printf("\n\tTop : %d", top);//3
   printf("\n\tValue: %d",stack[top]);//stack[3]=40
}
void change(int i, int new_element){
   stack[top-i+1] = new_element;

}

void main()
{
int ch;
int item, row, new_element;
clrscr();
while(1)
{
printf("\n*** Stack Menu ***");
printf("\n\n1.Push\n2.Pop\n3.display\n4.peep\n5.change\n6.exit"); printf("\n\nEnter your
choice(1-4):"); scanf("%d",&ch);
switch(ch)
{
```

```
case 1: push();
break;
case 2: pop();
break;
case 3: display();
break;
case 4:
          peep();
          break;
        case 5:
          printf("\n\tEnter row no : ");
          scanf("%d",&row);
          printf("\n\tEnter new element: ");
          scanf("%d", &new_element);
          change(row, new_element );
          break;
        case 6: exit(0);
break;
default: printf("\nWrong Choice!!");

}
}
getch();
}
```

OUTPUT:



```
*** Stack Menu ***

1.Push
2.Pop
3.display
4.peep
5.change
6.exit
Enter your choice(1-4):1
Enter element to push:56

*** Stack Menu ***

1.Push
2.Pop
3.display
4.peep
5.change
6.exit
Enter your choice(1-4):1
Enter element to push:89

*** Stack Menu ***

1.Push
2.Pop
3.display
4.peep
5.change
6.exit
Enter your choice(1-4):3

Stack is...
89
56

*** Stack Menu ***

1.Push
2.Pop
3.display
4.peep
5.change
```



```
56

*** Stack Menu ***

1.Push
2.Pop
3.display
4.peep
5.change
6.exit
Enter your choice(1-4):4

        Top : 1
        Value: 89
*** Stack Menu ***

1.Push
2.Pop
3.display
4.peep
5.change
6.exit
Enter your choice(1-4):5

        Enter row no : 2

        Enter new element: 22

*** Stack Menu ***

1.Push
2.Pop
3.display
4.peep
5.change
6.exit
Enter your choice(1-4):3

Stack is...
89
22

*** Stack Menu ***
```

Practical – 5

AIM: Implement a program to convert infix notation to postfix notation using stack.

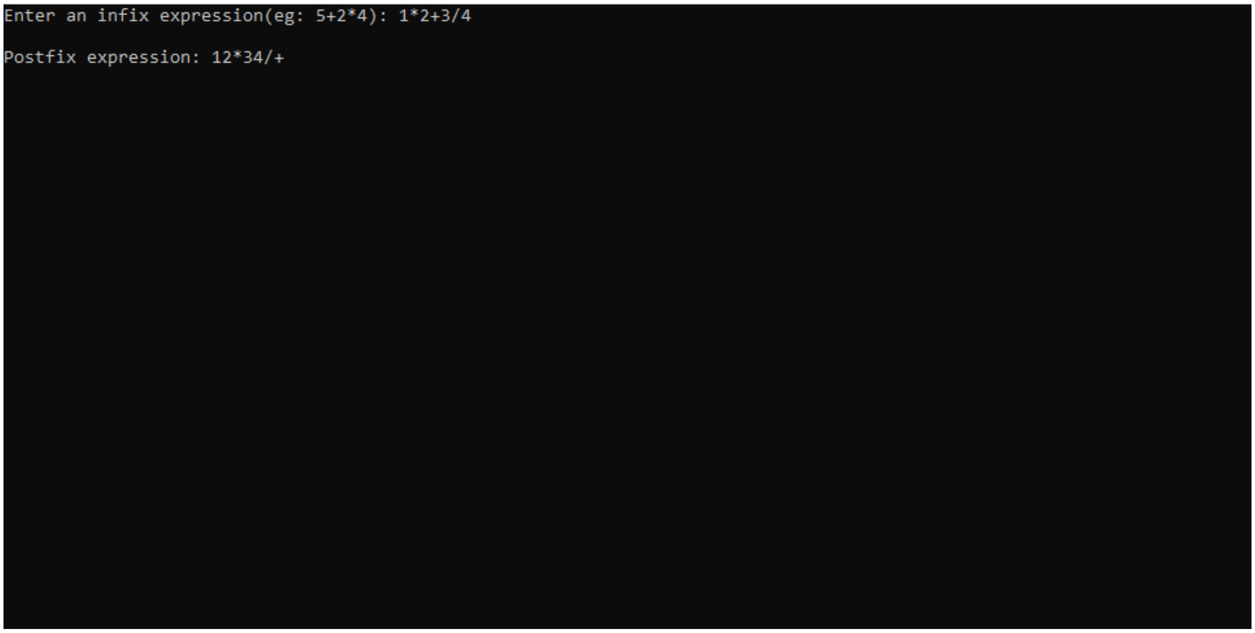
PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>//support isalnum() i.e for alpha numeric character

#define MAX 50

typedef struct stack
{
   int data[MAX];
   int top;
}stack;

int precedence(char);
void init(stack *);
int empty(stack *);
int full(stack *);
int pop(stack *);
void push(stack *,int);
int top(stack *);   //value of the top element
void infix_to_postfix(char infix[],char postfix[]);

void main()
{
   char infix[30],postfix[30];
   clrscr();
   printf("Enter an infix expression(eg: 5+2*4): ");
   gets(infix);
   infix_to_postfix(infix,postfix);
   printf("\nPostfix expression: %s",postfix);
   getch();
}

void infix_to_postfix(char infix[],char postfix[])
{
    stack s;
    char x,token;
    int i,j;   //i-index of infix,j-index of postfix
   // init(&s);
 s.top=-1;
    j=0;

    for(i=0;infix[i]!='\0';i++)
    {
            token=infix[i];
            if(isalnum(token))
                postfix[j++]=token;
            else
                if(token=='(')
                   push(&s,'(');
            else
                if(token==')')
                        while((x=pop(&s))!='(')
                              postfix[j++]=x;
                        else
                        {
                            while(precedence(token)<=precedence(top(&s))&&!empty(&s))
                            {
                                    x=pop(&s);
                                    postfix[j++]=x;
                            }
                            push(&s,token);
                        }

    }

    while(!empty(&s))
    {
            x=pop(&s);
            postfix[j++]=x;
    }

    postfix[j]='\0';
}
```

```
int precedence(char x)
{
    if(x=='(')
            return(0);
    if(x=='+'||x=='-')
            return(1);
    if(x=='*'||x=='/'||x=='%')
            return(2);

    return(0);
}

//void init(stack *s)
//{
  //  s->top=-1;
//}

int empty(stack *s)
{
    if(s->top==-1)
            return(1);

    return(0);
}

int full(stack *s)
{
    if(s->top==MAX-1)
        return(1);

    return(0);
}

void push(stack *s,int x)
{
    s->top=s->top+1;
    s->data[s->top]=x;
}

int pop(stack *s)
```

```
{
   int x;
   x=s->data[s->top];
   s->top=s->top-1;
   return(x);
}

int top(stack *p)
{
   return (p->data[p->top]);
}
```

RESULT:



```
Enter an infix expression(eg: 5+2*4): 1*2+3/4

Postfix expression: 12*34/+
```

Practical – 6

AIM: To implement simple queue using array and perform INSERT, DELETE and DISPLAY operations

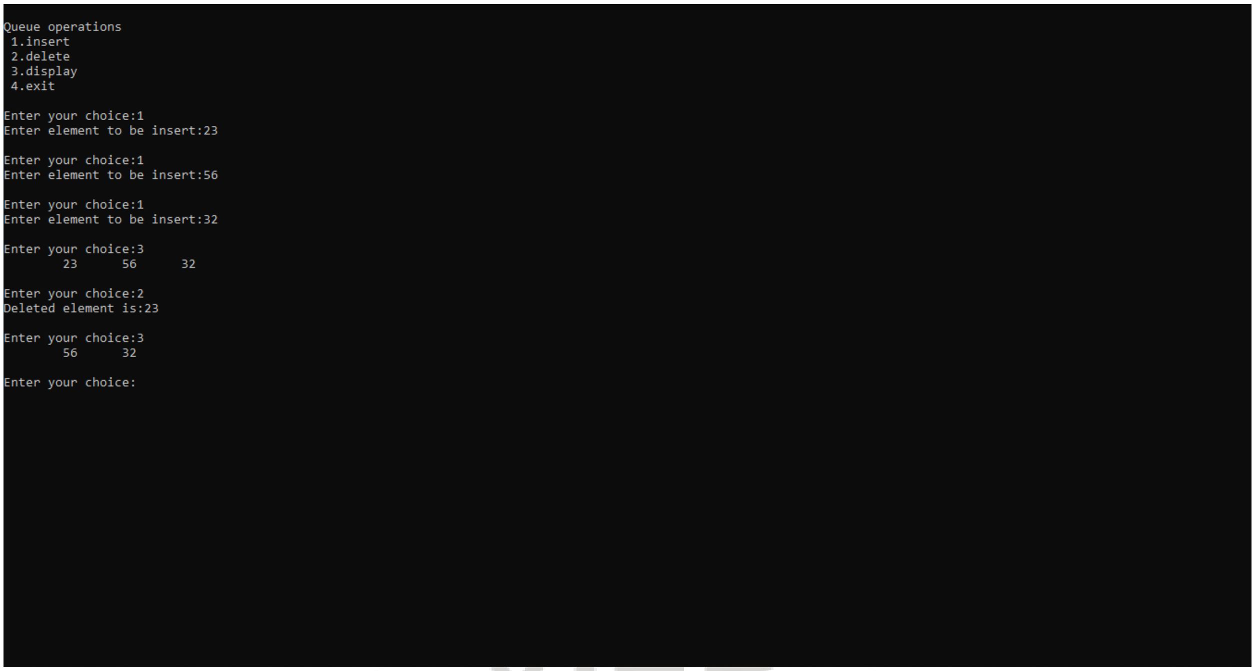
PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define max 10
int q[10],front=-1,rear=-1;
void insert();
void delet();
void display();
void main()
{
        int ch;
        clrscr();
        printf("\nQueue operations\n");
        printf("1.insert\n2.delete\n3.display\n4.exit\n");
        while(1)
        {
                printf("Enter your choice:"); scanf("%d",&ch);
                switch(ch)
                {
                        case 1:insert();
                        break;
                        case 2:delet();
                        break;
                        case 3:display();
                        break;
                        case 4:exit(0);
                        default:printf("Invalid option\n");
                }
        }
getch();
}
```

```
void insert()
{
        int x;
        if(rear==max-1)
        printf("Queue is overflow\n");
        else
        {
                if(front == -1)
                front=0;
                printf("Enter element to be insert:"); scanf("%d",&x);
                rear=rear+1;
                q[rear]=x;
        }
}
void delet()
        {
        int a;
        if((front==-1)&&(rear==-1))
        {
                printf("Queue is underflow\n");
        }
        a=q[front];
        front=front+1;
        printf("Deleted element is:%d\n",a);
        if(front>rear)
        {
                front=-1; rear=-1;//queue is empty
        }
}
void display()
{
        int i;
        if(front==-1 && rear==-1)
        {
                printf("Queue is underflow\n");
        }
        for(i=front;i<=rear;i++)
        {
                printf("\t%d",q[i]);
                printf("\n");
```

```
            }
getch();
}
```

OUTPUT:



```
Queue operations
 1.insert
 2.delete
 3.display
 4.exit

Enter your choice:1
Enter element to be insert:23

Enter your choice:1
Enter element to be insert:56

Enter your choice:1
Enter element to be insert:32

Enter your choice:3
        23      56      32

Enter your choice:2
Deleted element is:23

Enter your choice:3
        56      32

Enter your choice:
```

Practical – 7

AIM: To implement circular queue using array and perform INSERT, DELETE and DISPLAY operations.

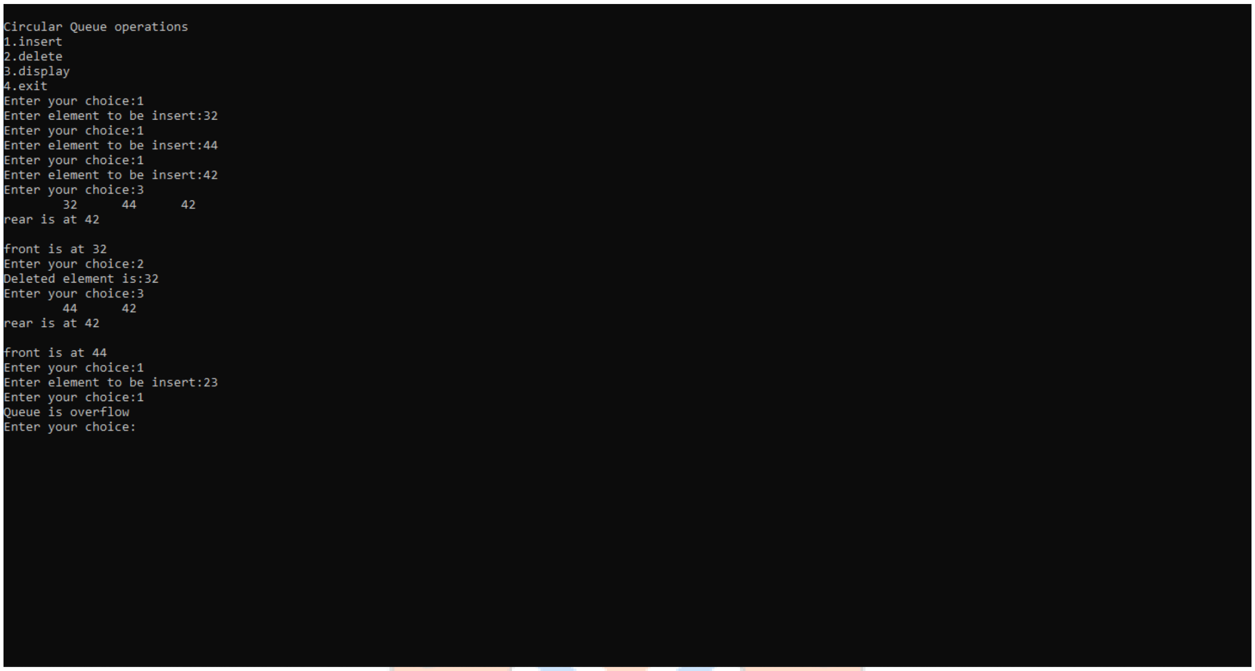
PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define max 10
int q[10],front=-1,rear=-1;
void insert();
void delet();
void display();
void main()
{
        int ch;
        clrscr();
        printf("\nQueue operations\n");
        printf("1.insert\n2.delete\n3.display\n4.exit\n");
        while(1)
        {
                printf("Enter your choice:"); scanf("%d",&ch);
                switch(ch)
                {
                        case 1:insert();
                        break;
                        case 2:delet();
                        break;
                        case 3:display();
                        break;
                        case 4:exit(0);
                        default:printf("Invalid option\n");
                }
        }
getch();
}
```

```
void insert()
{
    int x;
    if(rear==max-1)
        printf("Queue is overflow\n");
    else
    {
        if(front == -1)
        front=0;
        printf("Enter element to be insert:"); scanf("%d",&x);
        rear=rear+1;
        q[rear]=x;
    }
}
void delet()
{
    int a;
    if((front==-1)&&(rear==-1))
    {
        printf("Queue is underflow\n");
    }
    a=q[front];
    front=front+1;
    printf("Deleted element is:%d\n",a);
    if(front>rear)
    {
        front=-1; rear=-1;//queue is empty
    }
}

void display()
{
    int i;
    if(front==-1 && rear==-1)
    {
    printf("Queue is underflow\n");
    }
```

```
            for(i=front;i<=rear;i++)
            {
                        printf("\t%d",q[i]);
                        printf("\n");
            }
getch();
}
```

OUTPUT:



```
Circular Queue operations
1.insert
2.delete
3.display
4.exit
Enter your choice:1
Enter element to be insert:32
Enter your choice:1
Enter element to be insert:44
Enter your choice:1
Enter element to be insert:42
Enter your choice:3
        32      44      42
rear is at 42

front is at 32
Enter your choice:2
Deleted element is:32
Enter your choice:3
        44      42
rear is at 42

front is at 44
Enter your choice:1
Enter element to be insert:23
Enter your choice:1
Queue is overflow
Enter your choice:
```

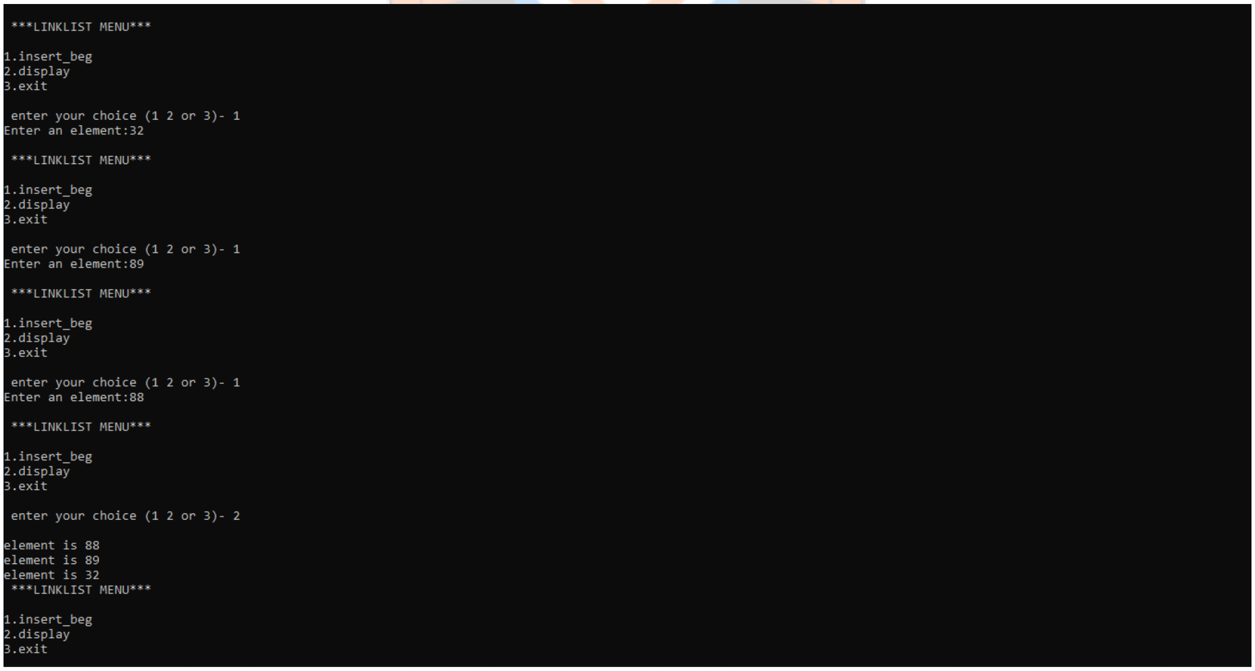
PRACTICAL – 8

AIM: Write a menu driven program to implement following operation on the singly linked list. a) Insert node at start of the linked list.

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
void insert_beg();

void display();

struct node
{
int data;
struct node *next;
};
struct node *start=NULL;

int main()
{
        int ch;
        for(;;)//infinity loop
        {
                printf("\n ***LINKLIST MENU***");
                printf("\n\n1.insert_beg\n2.display\n3.exit");
                printf("\n\n enter your choice (1 2 or 3)- ");
                scanf("%d",&ch);
                switch(ch)
                {
                case 1:insert_beg();
                     break;
                case 2:display();
                     break;
                case 3:exit(0);
                default:printf("\nwrong coice!");
                     break;
                        }
        }
}
void insert_beg()
{
        struct node *new_node;
        int val;
        new_node=(struct node*)(malloc(sizeof(struct node)));
                printf("Enter an element:");
                scanf("%d",&val);
        new_node->data=val;
        new_node->next=start;
        start=new_node;
}
 void display()
{
        struct node *ptr;
        ptr=start;
        while(ptr!=NULL)
{
        printf("\nelement is %d",ptr->data);
        ptr=ptr->next;
}
}
```

OUTPUT:



```
 ***LINKLIST MENU***

1.insert_beg
2.display
3.exit

 enter your choice (1 2 or 3)- 1
Enter an element:32

 ***LINKLIST MENU***

1.insert_beg
2.display
3.exit

 enter your choice (1 2 or 3)- 1
Enter an element:89

 ***LINKLIST MENU***

1.insert_beg
2.display
3.exit

 enter your choice (1 2 or 3)- 1
Enter an element:88

 ***LINKLIST MENU***

1.insert_beg
2.display
3.exit

 enter your choice (1 2 or 3)- 2

element is 88
element is 89
element is 32
 ***LINKLIST MENU***

1.insert_beg
2.display
3.exit
```

Practical – 9

AIM: Write a menu driven program to implement following operation on the singly linked list. a) Insert node at end of the linked list.

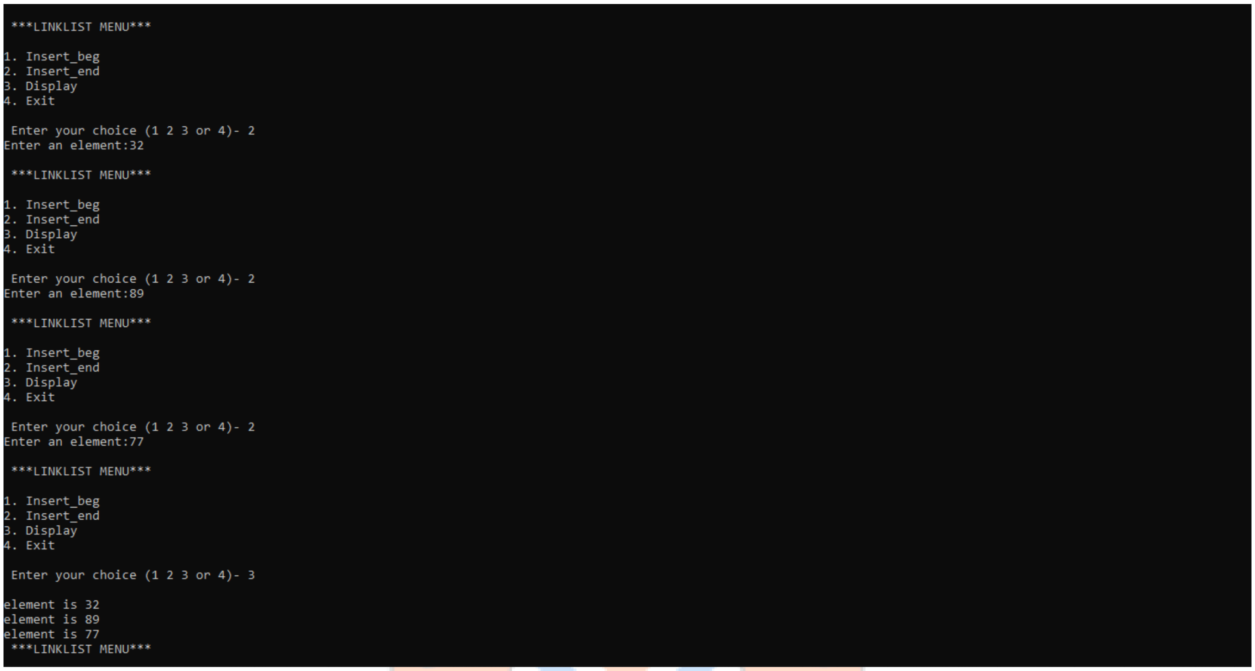
PROGRAM:

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

struct node * insert_beg();
struct node * insert_end();

void display();

struct node
{
        int data;
        struct node *next;
};
struct node *start=NULL;

int main()
{
        int ch;
        while(1)
        {
                printf("\n ***LINKLIST MENU***");
                printf("\n\n1. Insert_beg\n2. Insert_end\n3. Display\n4. Exit");
                printf("\n\n Enter your choice (1 2 3 or 4)- ");
                scanf("%d",&ch);
        switch(ch)
                {
                case 1:start=insert_beg(); break;
                case 2:start=insert_end(); break;
                case 3:display(); break;
                case 4:exit(0);
                break;
                 default:printf("\nwrong coice!");
                break;
                }
```

```
    }
}
struct node * insert_beg()
{
    struct node *new_node;
    int val;
    new_node=(struct node*)(malloc(sizeof(struct node))); printf("Enter an element:");
    scanf("%d",&val);
    new_node->data=val; new_node->next=start; start=new_node;
return start;
}


struct node * insert_end()
{
    struct node *new_node,*ptr;
    int val,i=1;
    new_node=(struct node*)(malloc(sizeof(struct node)));
    printf("Enter an element:");
    scanf("%d",&val);
    new_node->data=val;
    new_node->next=NULL;
    ptr=start;
    if(start==NULL)        //if link list is empty
    {
        start=new_node;
    }
    else
    {
        while(ptr->next!=NULL)
        {
        ptr=ptr->next;
        }
        ptr->next=new_node;
    }
    return start;
}
void display()
{
    struct node *ptr;
```

```
        ptr=start;
        while(ptr!=NULL)
        {
                printf("\nelement is %d",ptr->data);
                ptr=ptr->next;
        }
}
```

OUTPUT:



```
 ***LINKLIST MENU***

1. Insert_beg
2. Insert_end
3. Display
4. Exit

 Enter your choice (1 2 3 or 4)- 2
Enter an element:32

 ***LINKLIST MENU***

1. Insert_beg
2. Insert_end
3. Display
4. Exit

 Enter your choice (1 2 3 or 4)- 2
Enter an element:89

 ***LINKLIST MENU***

1. Insert_beg
2. Insert_end
3. Display
4. Exit

 Enter your choice (1 2 3 or 4)- 2
Enter an element:77

 ***LINKLIST MENU***

1. Insert_beg
2. Insert_end
3. Display
4. Exit

 Enter your choice (1 2 3 or 4)- 3

element is 32
element is 89
element is 77
 ***LINKLIST MENU***
```

Practical – 10

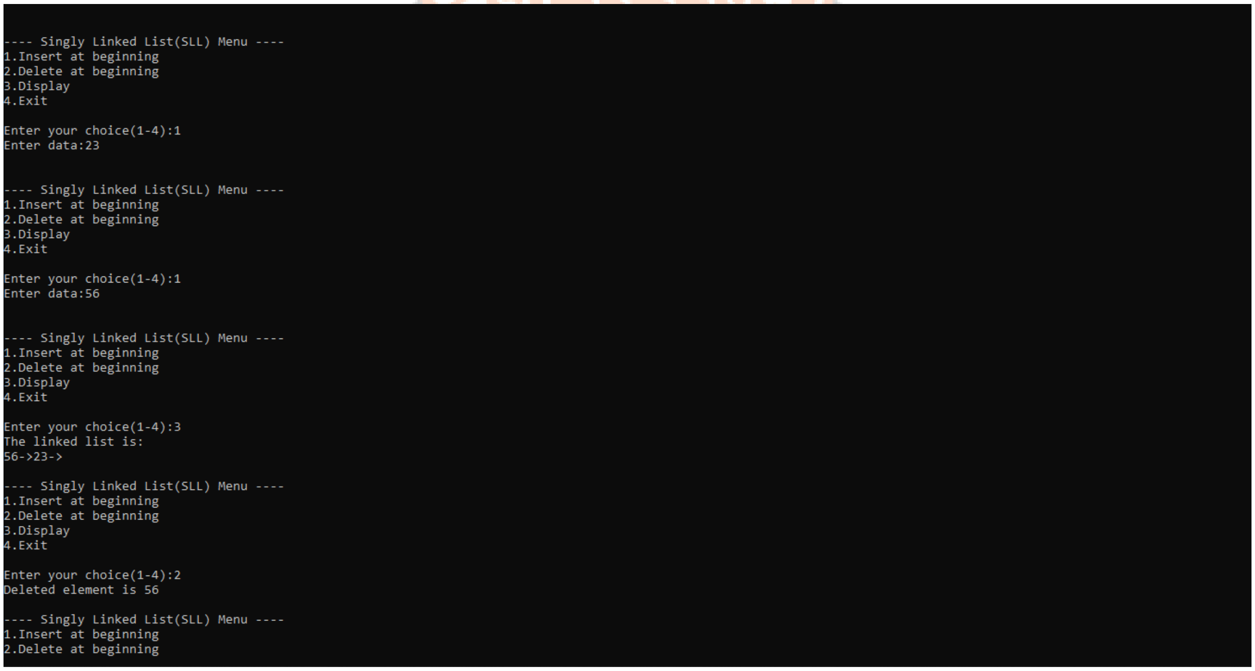
AIM: Write a menu driven program to implement following operation on the singly linked list. a) Delete first node of the linked list.

PROGRAM:

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

struct node
{
   int data;
   struct node *next;
}*start=NULL,*q,*new_node;
// struct node *start=null;
int main()
{
   int ch;
   void insert_beg();
   //void insert_end();

   void display();
   void delete_beg();
   while(1)
   {
      printf("\n\n---- Singly Linked List(SLL) Menu ----");
      printf("\n1.Insert at beginning\n2.Delete at beginning\n3.Display\n4.Exit\n\n");
      printf("Enter your choice(1-4):");
      scanf("%d",&ch);

      switch(ch)
      {
         case 1: insert_beg();
               break;
         case 2: delete_beg();
               break;
            case 3: display();
               break;
      case 4: exit(0);
               break;
           default: printf("Wrong Choice!!");
    }
  }
  return 0;
}

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

  if(start==NULL)    //If list is empty
  {
    new_node->next=NULL;
    start=new_node;
  }
  else
  {
    new_node->next=start;
    start=new_node;
  }
}


void delete_beg()
{
  if(start==NULL)
  {
    printf("The list is empty!!");
  }
  else
  {
    q=start;
    start=start->next;
    printf("Deleted element is %d",q->data);
```

```
        free(q);
    }
}

void display()
{
    if(start==NULL)
    {
        printf("List is empty!!");
    }
    else
    {
        q=start;
        printf("The linked list is:\n");
        while(q!=NULL)
        {
            printf("%d->",q->data);
              q=q->next;
        }
    }
}
```

OUTPUT:



```
---- Singly Linked List(SLL) Menu ----
1.Insert at beginning
2.Delete at beginning
3.Display
4.Exit

Enter your choice(1-4):1
Enter data:23


---- Singly Linked List(SLL) Menu ----
1.Insert at beginning
2.Delete at beginning
3.Display
4.Exit

Enter your choice(1-4):1
Enter data:56


---- Singly Linked List(SLL) Menu ----
1.Insert at beginning
2.Delete at beginning
3.Display
4.Exit

Enter your choice(1-4):3
The linked list is:
56->23->

---- Singly Linked List(SLL) Menu ----
1.Insert at beginning
2.Delete at beginning
3.Display
4.Exit

Enter your choice(1-4):2
Deleted element is 56

---- Singly Linked List(SLL) Menu ----
1.Insert at beginning
2.Delete at beginning
```

Practical – 11

AIM: Write a menu driven program to implement following operation on the singly linked list. a) Delete node before given node of the linked list.

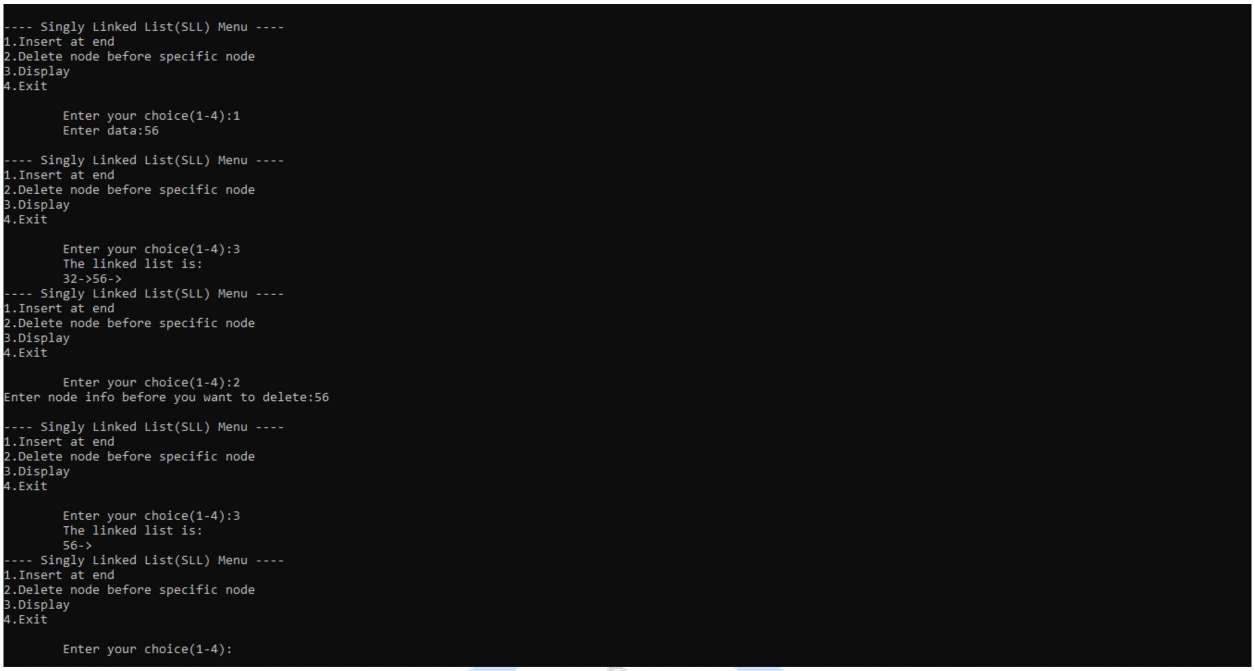
PROGRAM:

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

struct node
{
    int data;
    struct node *next;
}*start=NULL,*ptr,*new_node;
int main()
{
    int ch;
    void insert_end();
    void display();
    void del_before();

    while(1)
    {
            printf("\n\n---- Singly Linked List(SLL) Menu ----");
            printf("\n1.Insert at end\n2.Delete node before specific node\n3.Display\n4.Exit\n\n\t");
            printf("Enter your choice(1-4):");
            scanf("%d",&ch);

            switch(ch)
            {
                case 1: insert_end();
                            break;
                case 2: del_before();
                            break;

                case 3: display();
                            break;
                case 4: exit(0);
```

```
        default: printf("Wrong Choice!!");
              }
  }
}

void insert_end()
{
  int num;
  new_node=(struct node*)malloc(sizeof(struct node));
  printf("\tEnter data:");
  scanf("%d",&num);
  new_node->data=num;
  new_node->next=NULL;

  if(start==NULL)     //If list is empty
  {
            start=new_node;
  }
  else
  {
            ptr=start;
      while(ptr->next!=NULL)
            ptr=ptr->next;
            ptr->next=new_node;
  }
}


void display()
{
  if(start==NULL)
  {
            printf("List is empty!!");
  }
  else
  {
            ptr=start;
            printf("\tThe linked list is:\n\t");
      while(ptr!=NULL)
      {
```

```
     printf("%d->",ptr->data);
     ptr=ptr->next;
    }
 }
}
void del_before()
{
int info;
printf("Enter node info before you want to delete:");
scanf("%d",&info);
    struct node *t,*t2,*t3;
    t=start;
    if(info==start->data)
    {
        printf("\tNODE CANNOT BE DELETED\n");
    }
    else
    {
        if(info==start->next->data)
        {
            t3=start;
            start=start->next;
            free(t3);
        }
        else
        {
        while(t->next->next->data!=info && t->next->next!=NULL)
            {
                t=t->next;
            }
            if(t->next->next->data==info)
            {
                t2=t->next;
                t->next=t2->next;
                free(t2);
            }
        }
    }
}
```

OUTPUT:



```
---- Singly Linked List(SLL) Menu ----
1.Insert at end
2.Delete node before specific node
3.Display
4.Exit

        Enter your choice(1-4):1
        Enter data:56

---- Singly Linked List(SLL) Menu ----
1.Insert at end
2.Delete node before specific node
3.Display
4.Exit

        Enter your choice(1-4):3
        The linked list is:
        32->56->
---- Singly Linked List(SLL) Menu ----
1.Insert at end
2.Delete node before specific node
3.Display
4.Exit

        Enter your choice(1-4):2
Enter node info before you want to delete:56

---- Singly Linked List(SLL) Menu ----
1.Insert at end
2.Delete node before specific node
3.Display
4.Exit

        Enter your choice(1-4):3
        The linked list is:
        56->
---- Singly Linked List(SLL) Menu ----
1.Insert at end
2.Delete node before specific node
3.Display
4.Exit

        Enter your choice(1-4):
```

Practical – 12

AIM: Write a menu driven program to implement following operation on the singly linked list. a) Delete node after given node of the linked list.

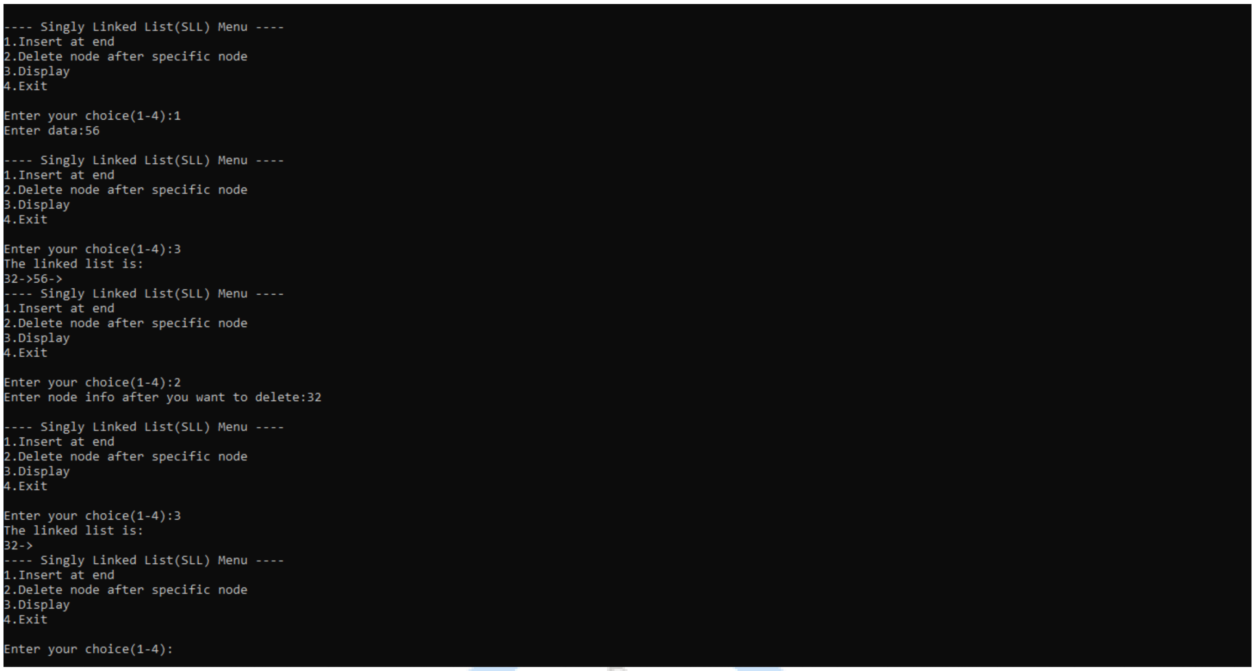
PROGRAM:

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

struct node
{
    int data;
    struct node *next;
}*start=NULL,*ptr,*new_node;

int main()
{
    int ch;
    void insert_end();
    void display();
    void del_after();

    while(1)
    {
            printf("\n\n---- Singly Linked List(SLL) Menu ----");
            printf("\n1.Insert at end\n2.Delete node after specific node\n3.Display\n4.Exit\n\n");
            printf("Enter your choice(1-4):");
            scanf("%d",&ch);

            switch(ch)
            {
                case 1: insert_end();
                        break;
                case 2: del_after();
                        break;

                case 3: display();
```

```
                        break;
                case 4: exit(0);
                default: printf("Wrong Choice!!");
                        }
    }
}

void insert_end()
{
    int num;
    new_node=(struct node*)malloc(sizeof(struct node));
    printf("Enter data:");
    scanf("%d",&num);
    new_node->data=num;
    new_node->next=NULL;

    if(start==NULL)        //If list is empty
    {
            start=new_node;
    }
    else
    {
            ptr=start;
            while(ptr->next!=NULL)
            ptr=ptr->next;
            ptr->next=new_node;
    }
}


void display()
{
    if(start==NULL)
    {
            printf("List is empty!!");
    }
    else
    {
            ptr=start;
            printf("The linked list is:\n");
```

```
        while(ptr!=NULL)
        {
           printf("%d->",ptr->data);
           ptr=ptr->next;
        }
   }
}
void del_after()
{
int info;
printf("Enter node info after you want to delete:");
scanf("%d",&info);
        struct node *t,*ptr,*t1;
        ptr=start;
        if(info==start->data)
        {
                t=start->next;
                start->next=t->next;
                free(t);
        }
        while(ptr->next!=NULL)
        {
                ptr=ptr->next;

        if(ptr->data==info)
        {
                t1=ptr->next;
                if(t1->next==NULL)
                {
                                ptr->next=NULL;
                }
                else
                {
                        ptr->next=t1->next;
                }
                free(t1);
                }
        }

}
```

OUTPUT:



```
---- Singly Linked List(SLL) Menu ----
1.Insert at end
2.Delete node after specific node
3.Display
4.Exit

Enter your choice(1-4):1
Enter data:56

---- Singly Linked List(SLL) Menu ----
1.Insert at end
2.Delete node after specific node
3.Display
4.Exit

Enter your choice(1-4):3
The linked list is:
32->56->
---- Singly Linked List(SLL) Menu ----
1.Insert at end
2.Delete node after specific node
3.Display
4.Exit

Enter your choice(1-4):2
Enter node info after you want to delete:32

---- Singly Linked List(SLL) Menu ----
1.Insert at end
2.Delete node after specific node
3.Display
4.Exit

Enter your choice(1-4):3
The linked list is:
32->
---- Singly Linked List(SLL) Menu ----
1.Insert at end
2.Delete node after specific node
3.Display
4.Exit

Enter your choice(1-4):
```

Practical – 13

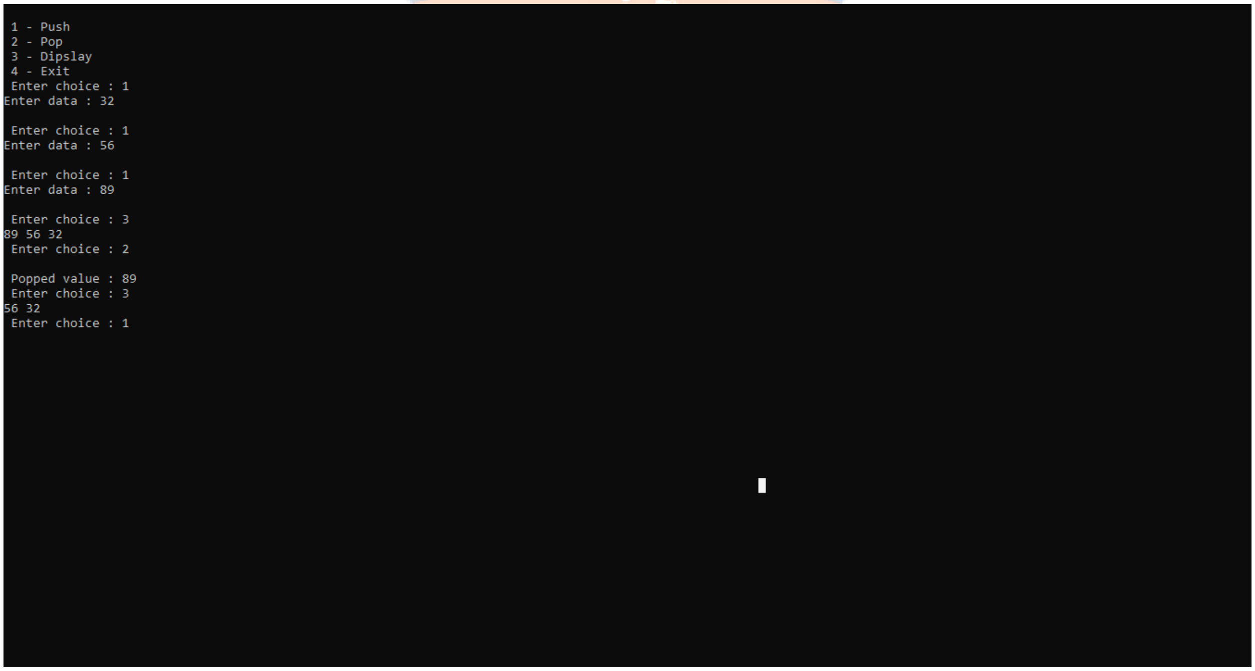
AIM: Write a program to implement stack using linked list.

PROGRAM:

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

struct node
{
        int info;
        struct node *ptr;
}*top=NULL,*top1,*temp;

void push(int data);
void pop();
void display();

void main()
{
        int no, ch, e;

        printf("\n 1 - Push");
        printf("\n 2 - Pop");
        printf("\n 3 - Dipslay");
        printf("\n 4 - Exit");

        while (1)
        {
        printf("\n Enter choice : "); scanf("%d", &ch);

        switch (ch)
        {
                case 1:
                printf("Enter data : ");
                scanf("%d", &no); push(no);
                break;
                case 2:
                pop();
                break;
        case 3:
        display();
        break;
        case 4:
        exit(0);
        default :
        printf(" Wrong choice, Please enter correct choice "); break;
        }
    }
}

/* Push data into stack */
void push(int data)
{
    if (top == NULL)
    {
        top =(struct node *)malloc(1*sizeof(struct node));
        top->ptr = NULL;
        top->info = data;
    }
    else
    {
        temp =(struct node *)malloc(1*sizeof(struct node));
        temp->ptr = top;
        temp->info = data;
        top = temp;
    }
}

/* Display stack elements */
void display()
{
    top1 = top;

    if (top1 == NULL)
    {
        printf("Stack is empty"); return;
    }

    while (top1 != NULL)
```

```
        {
                printf("%d ", top1->info); top1 = top1->ptr;
        }
}

/* Pop Operation on stack */
void pop()
{
        top1 = top;
        if (top1 == NULL)
        {
                printf("\n Error : Trying to pop from empty stack"); return;
        }
        else
        top1 = top1->ptr;
        printf("\n Popped value : %d", top->info);

        free(top); top = top1;
}
```

OUTPUT:



```
 1 - Push
 2 - Pop
 3 - Dipslay
 4 - Exit
 Enter choice : 1
Enter data : 32

 Enter choice : 1
Enter data : 56

 Enter choice : 1
Enter data : 89

 Enter choice : 3
89 56 32
 Enter choice : 2

 Popped value : 89
 Enter choice : 3
56 32
 Enter choice : 1
```

Practical – 14

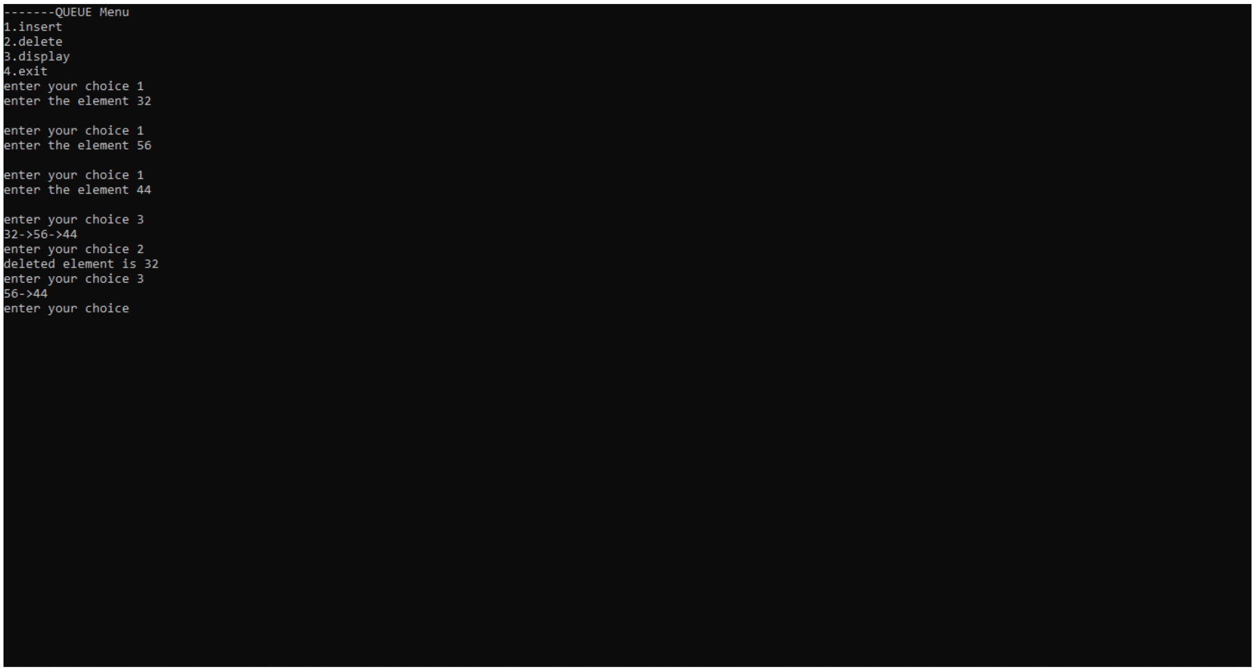
AIM: Write a program to implement queue using linked list.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
        int data;
        struct node *next;
}*f=NULL,*r=NULL,*ptr,*newnode;

int ele,a;
void insert();
void delete1();
void display();

int main()
{
        int x;
        printf("-------QUEUE Menu  ");
        printf("\n1.insert \n2.delete \n3.display \n4.exit");
        while(1){
        printf("\nenter your choice ");
        scanf("%d",&x);

        switch(x)
        {
                case 1: insert();break;
                case 2: delete1();break;
                case 3: display();break;
                case 4: exit(0);break;
                default :
                printf(" Wrong choice, Please enter correct choice "); break;
                }
        }
}
void insert()
```

```
{
        printf("enter the element ");
        scanf("%d",&ele);

        newnode=(struct node*)malloc(sizeof (struct node)); newnode->data=ele;
        newnode->next=NULL;

        if(r==NULL)
        {
                r=newnode; f=r;
        }
        else
        {
                r->next=newnode;
                r=newnode;
        }
}


void display()
{
        if(f==NULL)
        {
                printf("link list is empty");
        }
        else
        {
                ptr=f;
        while(ptr->next!=NULL)
        {
                printf("%d->",ptr->data);
                ptr=ptr->next;
        }
        printf("%d",ptr->data);
        }
}
void delete1()
{
        if(f==NULL)
        {
                printf("linklist is overflow");
        }
        else
        {
                ptr=f;
                f=f->next;
                printf("deleted element is %d",ptr->data);
                free (ptr);
        }
}
```

OUTPUT:



```
-------QUEUE Menu
1.insert
2.delete
3.display
4.exit
enter your choice 1
enter the element 32

enter your choice 1
enter the element 56

enter your choice 1
enter the element 44

enter your choice 3
32->56->44
enter your choice 2
deleted element is 32
enter your choice 3
56->44
enter your choice
```

Practical – 15

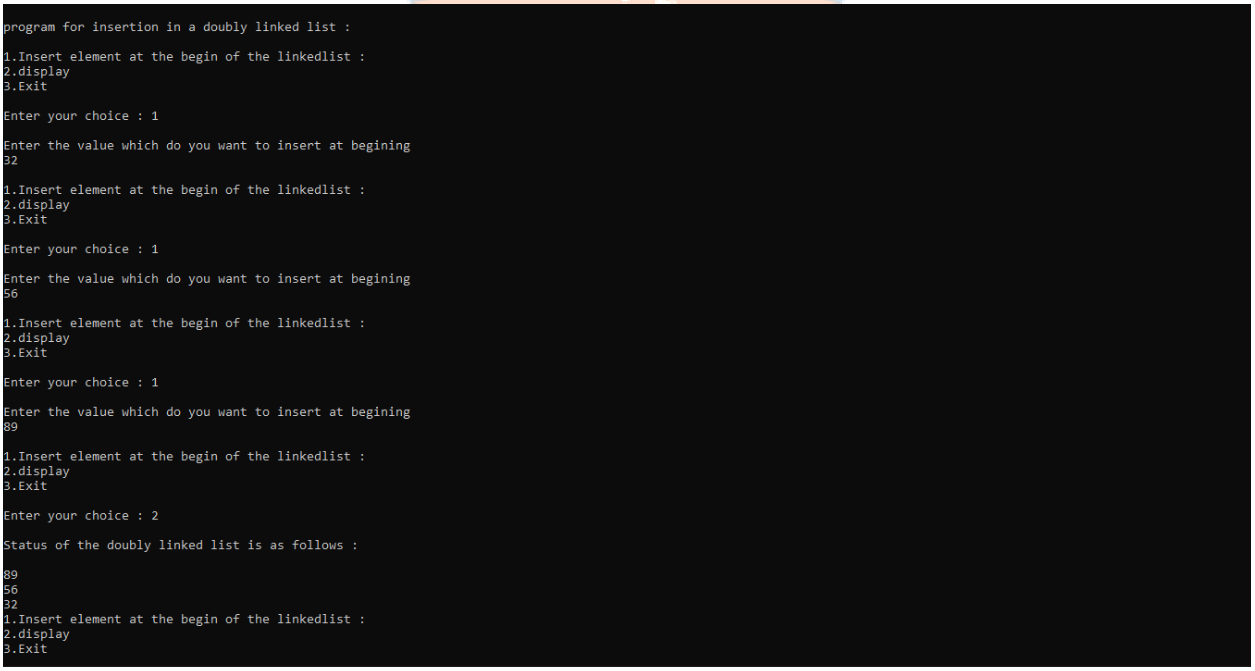
AIM: Write a menu driven program to implement following operation on the doubly linked list. a) Insert a node at the front of the doubly linked list.

PROGRAM:

```
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
struct node{
        int num;
        struct node *next;
        struct node *prev;
};
struct node *head=NULL,*temp, *first, *last;

int info;
void display();
void insert_at_begin();

int main()
        {
        int i;
        printf("\nprogram for insertion in a doubly linked list :\n");
        do {
                printf("\n1.Insert element at the begin of the linkedlist :");
                printf("\n2.display"); printf("\n3.Exit\n");
                printf("\nEnter your choice : ");
                scanf("%d",&i);
                switch(i) {
                        case 1: insert_at_begin();
                        break;
                        case 2:
                        display();
                        break; case 3: exit(0);
                }
        } while(1);

}
```

```
void display() {
        struct node *ptr; ptr=head;
        printf("\nStatus of the doubly linked list is as follows :\n");
        while(ptr!=NULL) /* traversing the linked list */
        { printf("\n%d",ptr->num); ptr=ptr->next; }
}
void insert_at_begin() {
        printf("\nEnter the value which do you want to insert at begining\n");
        scanf("%d",&info);
        temp=(struct node *)malloc(sizeof(struct node));
        //(struct node)malloc(sizeof(NODE));
        temp->num=info; temp->next=NULL;
        temp->prev=NULL;
        if(head==NULL) { head=temp; last=temp; }
        else {
                temp->next=head; head->prev=temp;
                temp->prev=NULL; head=temp;
        }
}
```

OUTPUT:



```
program for insertion in a doubly linked list :

1.Insert element at the begin of the linkedlist :
2.display
3.Exit

Enter your choice : 1

Enter the value which do you want to insert at begining
32

1.Insert element at the begin of the linkedlist :
2.display
3.Exit

Enter your choice : 1

Enter the value which do you want to insert at begining
56

1.Insert element at the begin of the linkedlist :
2.display
3.Exit

Enter your choice : 1

Enter the value which do you want to insert at begining
89

1.Insert element at the begin of the linkedlist :
2.display
3.Exit

Enter your choice : 2

Status of the doubly linked list is as follows :

89
56
32
1.Insert element at the begin of the linkedlist :
2.display
3.Exit
```

Practical – 16

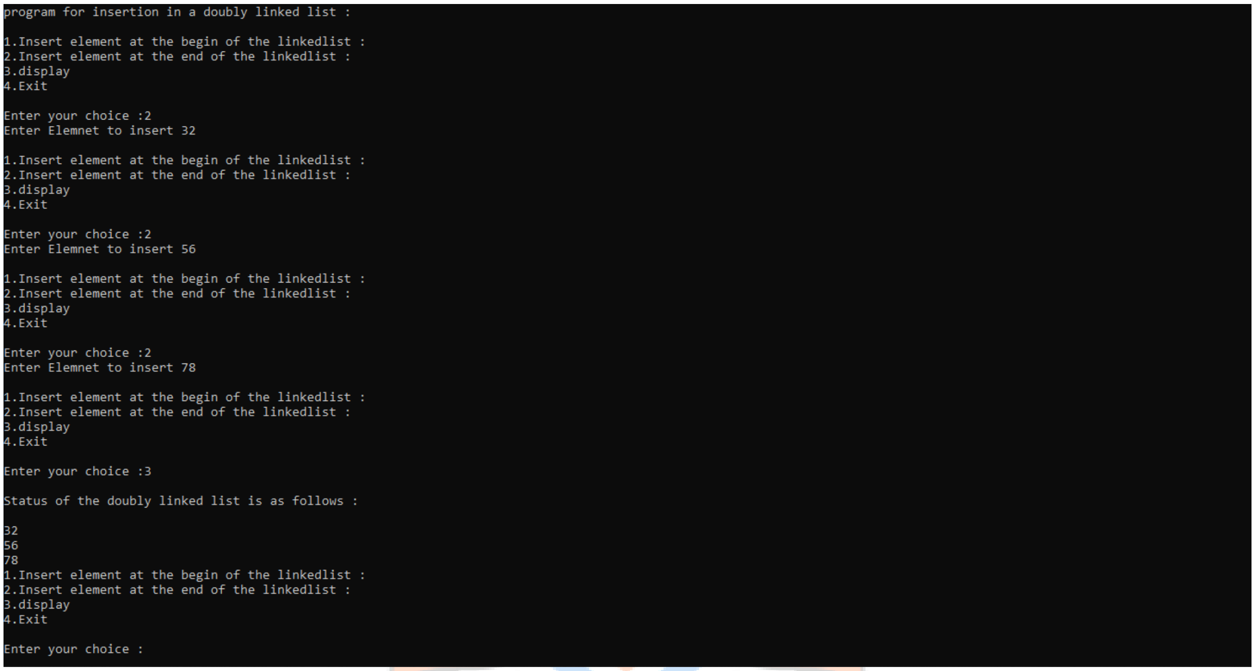
AIM: Write a menu driven program to implement following operation on the doubly linked list. a) Insert a node at the end of the doubly linked list.

PROGRAM:

```
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
struct node
{       int num;
        struct node *next;
        struct node *prev;
};

struct node *head=NULL,*temp, *first, *last;
int info;
void display();
void insert_at_begin();
void insert_at_end();

int main() {
        int i;
        printf("\nprogram for insertion in a doubly linked list :\n");
        do {
                printf("\nYASH PATIL Enter your choice :\n");
                printf("\n1.Insert element at the begin of the linkedlist :");
                printf("\n2.Insert element at the end of the linkedlist :");
                printf("\n3.display"); printf("\n4.Exit\n");
                scanf("%d",&i);
                switch(i) {
                        case 1: insert_at_begin();
                        break;
                        case 2:
                        insert_at_end();
                        break;
                        case 3:
                        display();
                        break;case 4: exit(0);
        }
    } while(1);
}
void display() {
    struct node *ptr; ptr=head;
    printf("\nStatus of the doubly linked list is as follows :\n");
    while(ptr!=NULL) /* traversing the linked list */
    {
        printf("\n%d",ptr->num); ptr=ptr->next;
    }
}
void insert_at_begin() {
        printf("\nEnter the value which do you want to insert at begining\n");
        scanf("%d",&info);
        temp=(struct node *)malloc(sizeof(struct node));
        //(struct node)malloc(sizeof(NODE));
        temp->num=info; temp->next=NULL;
        temp->prev=NULL;
        if(head==NULL) {
            head=temp; last=temp;
        }
        else {
            temp->next=head; head->prev=temp;
            temp->prev=NULL; head=temp;
        }
}

void insert_at_end(){
    struct node *ptr;
    printf("\nEnter Elemnet to insert ");
    scanf("%d",&info);
    temp=(struct node *)malloc(sizeof(struct node));
    temp->num=info;
    temp->next=NULL;
    temp->prev=NULL;
    if(head==NULL){
        head=temp;last=temp;
    }
    ptr=head;
    while(ptr->next!=NULL){
```

```
                ptr=ptr->next;
        }
        ptr->next=temp;
        temp->prev=ptr;
        temp->next=NULL;
}
```

OUTPUT:



```
program for insertion in a doubly linked list :

1.Insert element at the begin of the linkedlist :
2.Insert element at the end of the linkedlist :
3.display
4.Exit

Enter your choice :2
Enter Elemnet to insert 32

1.Insert element at the begin of the linkedlist :
2.Insert element at the end of the linkedlist :
3.display
4.Exit

Enter your choice :2
Enter Elemnet to insert 56

1.Insert element at the begin of the linkedlist :
2.Insert element at the end of the linkedlist :
3.display
4.Exit

Enter your choice :2
Enter Elemnet to insert 78

1.Insert element at the begin of the linkedlist :
2.Insert element at the end of the linkedlist :
3.display
4.Exit

Enter your choice :3

Status of the doubly linked list is as follows :

32
56
78
1.Insert element at the begin of the linkedlist :
2.Insert element at the end of the linkedlist :
3.display
4.Exit

Enter your choice :
```

Practical – 17

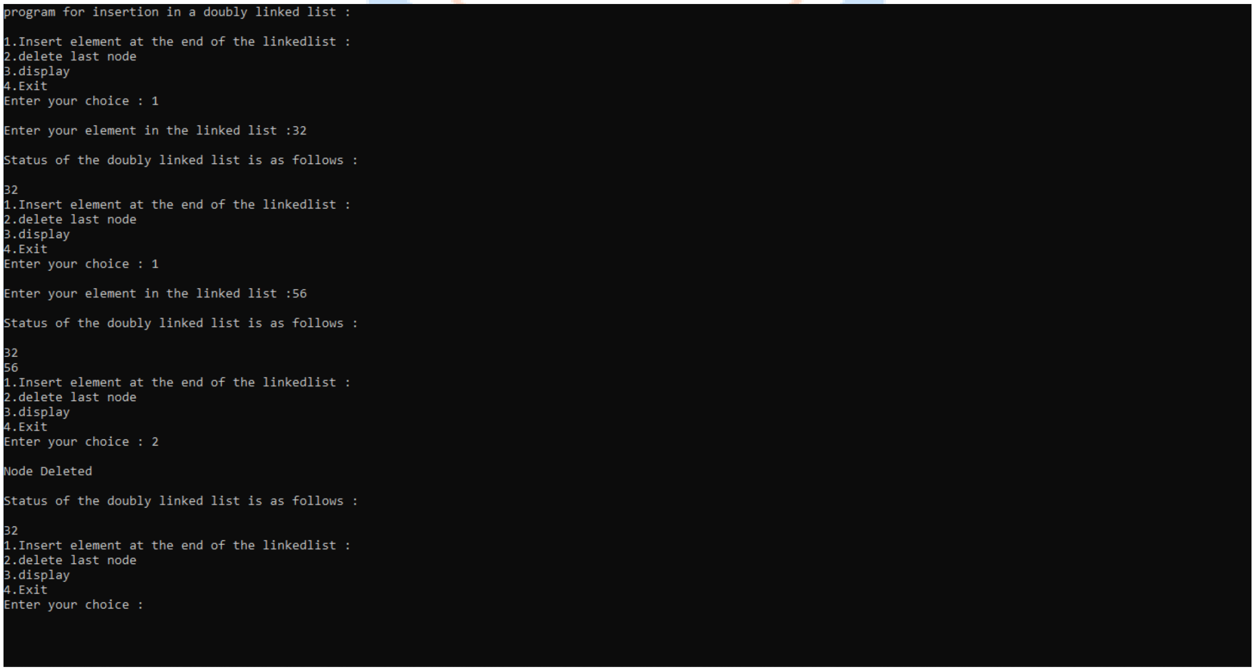
AIM: Write a menu driven program to implement following operation on the doubly linked list. a) Delete last node of the doubly linked list.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
//#include<process.h>
struct node{
        int num;
        struct node *next;
        struct node *prev;
};
struct node *head=NULL,*temp, *first, *last;

int info;
void insert_at_end();
void display();
void del_at_end();
int main() /* starting the main method() */
{
        int i;
        printf("program for insertion in a doubly linked list :\n");
        do {
        printf("\n1.Insert element at the end of the linkedlist :");
        printf("\n2.delete last node");
        printf("\n3.display");
        printf("\n4.Exit\n");
        printf("Enter your choice : ");
        scanf("%d",&i);
        switch(i) {
                case 1:
                insert_at_end();
                display();
                break;
                case 2:
                del_at_end();
        display();
        break;
        case 3:
        display();
        break;
        case 4: exit(0);
        }
    }
    while(1);
}
void display() {
    struct node *ptr;
    ptr=head;
    printf("\nStatus of the doubly linked list is as follows :\n");
    while(ptr!=NULL) /* traversing the linked list */
    {
        printf("\n%d",ptr->num); ptr=ptr->next;
    }
}
void insert_at_end(){
    struct node *ptr; printf("\nEnter your element in the linked list :"); scanf("%d",&info);
    temp=(struct node *)malloc(sizeof(struct node)); /* allocating memory for the node to be
inserted */
    temp->num=info;
    temp->next=NULL;
    temp->prev=NULL;
    if(head==NULL) { head=temp; last=temp; }
    ptr=head;
    while(ptr->next!=NULL)
    { ptr=ptr->next;
    }
    ptr->next=temp; temp->prev=ptr; temp->next=NULL;
}
void del_at_end()
{
    struct node * ptr;
    if(head == NULL)
    {
        printf(" Delete is not possible. No data in the list.\n");
    }
```

```
        else if(head->next == NULL)
        {
                head = NULL;
                free(head);
                printf("\nNode Deleted\n");
        }
        else
        {
                ptr = head;
        while(ptr->next != NULL)
        {
                ptr = ptr -> next;
        }
        ptr -> prev -> next = NULL;
        free(ptr);
        printf("\nNode Deleted\n");
        }
}
```

OUTPUT:



```
program for insertion in a doubly linked list :

1.Insert element at the end of the linkedlist :
2.delete last node
3.display
4.Exit
Enter your choice : 1

Enter your element in the linked list :32

Status of the doubly linked list is as follows :

32
1.Insert element at the end of the linkedlist :
2.delete last node
3.display
4.Exit
Enter your choice : 1

Enter your element in the linked list :56

Status of the doubly linked list is as follows :

32
56
1.Insert element at the end of the linkedlist :
2.delete last node
3.display
4.Exit
Enter your choice : 2

Node Deleted

Status of the doubly linked list is as follows :

32
1.Insert element at the end of the linkedlist :
2.delete last node
3.display
4.Exit
Enter your choice :
```

Practical – 18

AIM: Write a menu driven program to implement following operation on the doubly linked list. a) Delete a node after a specified position in the doubly linked list.

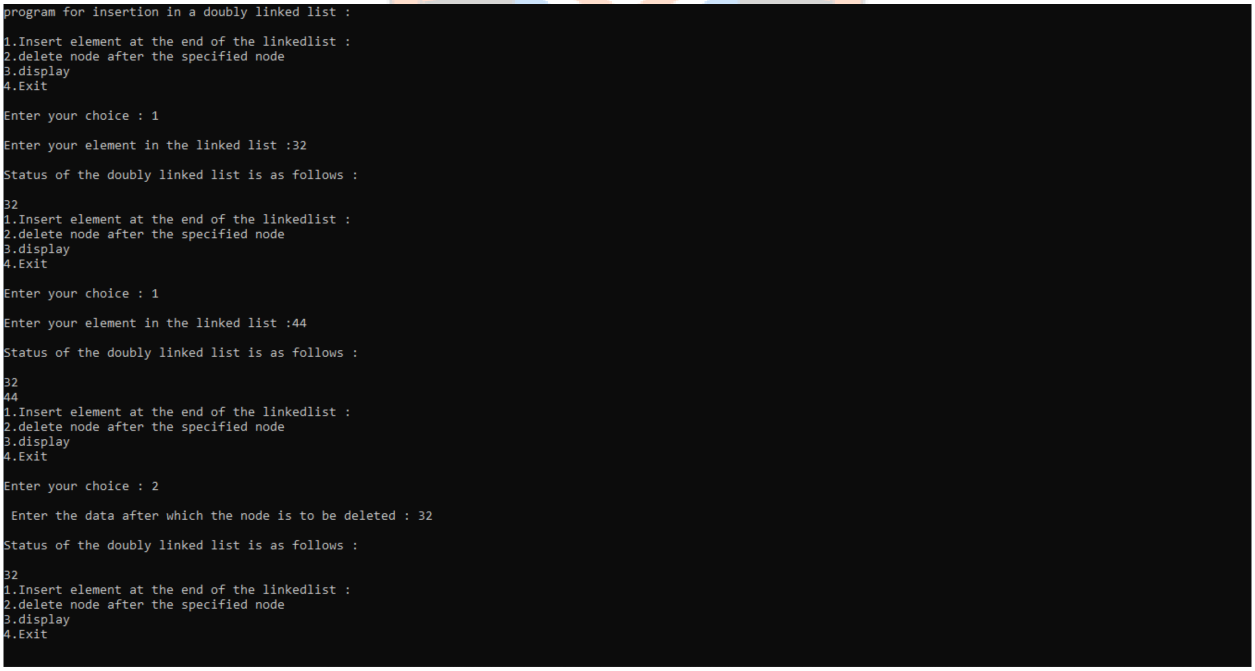
PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
//#include<process.h>
struct node{
    int num;
    struct node *next;
    struct node *prev;
};
struct node *head=NULL,*temp, *first, *last;

int info;
void del_after_pos();
void display();
void insert_at_end();

int main() { /* starting the main method() */
    int i;
    printf("\nprogram for insertion in a doubly linked list :\n");
    do {
        printf("\nEnter your choice :\n");
        printf("\n1.Insert element at the end of the linkedlist :");
        printf("\n2.delete node after the specified node");
        printf("\n3.display");
        printf("\n4.Exit\n");
        scanf("%d",&i);
        switch(i) {
            case 1:
            insert_at_end();
            display();
            break;
            case 2:
```

```
            del_after_pos();
            display();
            break;
            case 3:
            display();
            break;
            case 4: exit(0);
        }
    }
    while(1);
 }
void display() {
    struct node *ptr;
    ptr=head;
    printf("\nStatus of the doubly linked list is as follows :\n");
    while(ptr!=NULL) /* traversing the linked list */
    { printf("\n%d",ptr->num); ptr=ptr->next; }
}
void insert_at_end(){
    struct node *ptr; printf("\nEnter your element in the linked list :"); scanf("%d",&info);
    temp=(struct node *)malloc(sizeof(struct node)); /* allocating memory for the node to be
inserted */
    temp->num=info;
    temp->next=NULL;
    temp->prev=NULL;
    if(head==NULL) { head=temp; last=temp; }
    ptr=head;
    while(ptr->next!=NULL)
    { ptr=ptr->next;
    }
    ptr->next=temp; temp->prev=ptr; temp->next=NULL;
}
void del_after_pos()
{
    struct node *ptr, *temp;
    if( head == NULL)
    { printf("list is empty"); }
    else{
        int val;
        printf("\n Enter the data after which the node is to be deleted : ");
```

```
                scanf("%d", &val);
                ptr = head;
                while(ptr -> num != val)
                        ptr = ptr -> next;
                if(ptr -> next == NULL)
                {
                        printf("\nCan't delete\n");
                }
                else if(ptr -> next -> next == NULL)
                {
                        ptr ->next = NULL;
                }
                else
                {
                        temp = ptr -> next;
                        ptr -> next = temp -> next;
                        temp -> next -> prev = ptr;
                        free(temp);
                        printf("\nnode deleted\n");
                }
        }
}
```

OUTPUT:



```
program for insertion in a doubly linked list :

1.Insert element at the end of the linkedlist :
2.delete node after the specified node
3.display
4.Exit

Enter your choice : 1

Enter your element in the linked list :32

Status of the doubly linked list is as follows :

32
1.Insert element at the end of the linkedlist :
2.delete node after the specified node
3.display
4.Exit

Enter your choice : 1

Enter your element in the linked list :44

Status of the doubly linked list is as follows :

32
44
1.Insert element at the end of the linkedlist :
2.delete node after the specified node
3.display
4.Exit

Enter your choice : 2

 Enter the data after which the node is to be deleted : 32

Status of the doubly linked list is as follows :

32
1.Insert element at the end of the linkedlist :
2.delete node after the specified node
3.display
4.Exit
```

Practical – 19

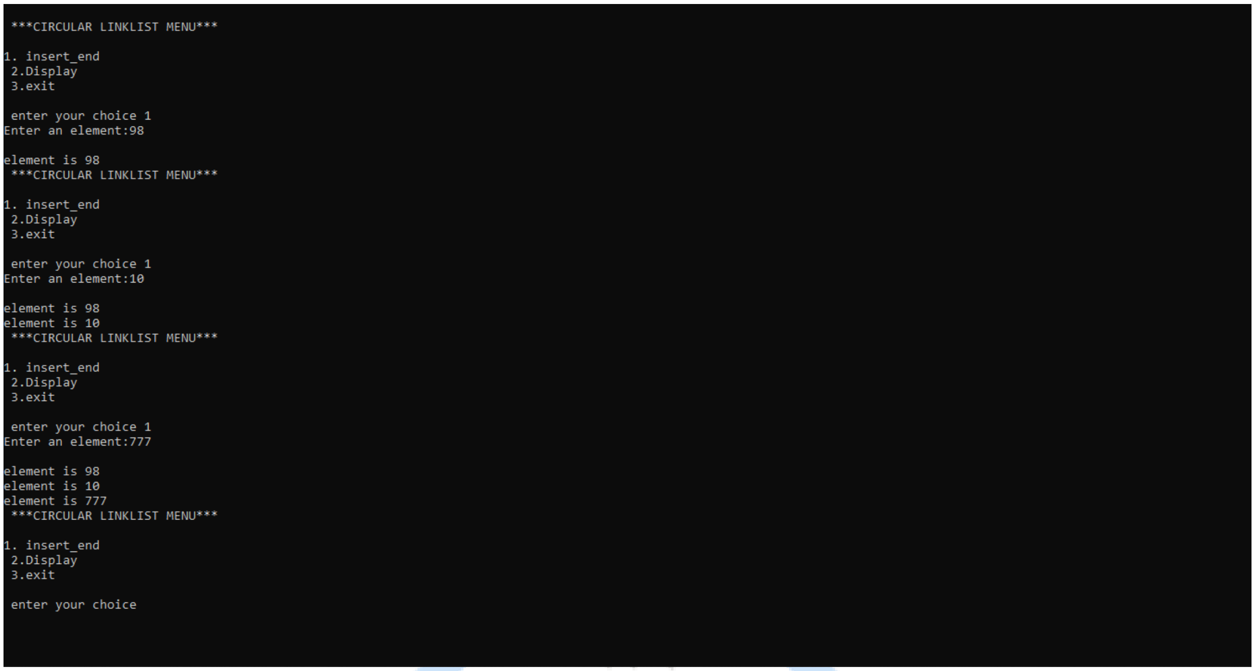
AIM: Write a program to implement the following operation on circular linked list

a) Insert node at end

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
//void insert_beg();
void insert_end();
void display();
struct node {
        int data;
        struct node *next;
}*start=NULL;


int main() {
int ch;
        while(1)
        {
                printf("\n ***CIRCULAR LINKLIST MENU***");
                printf("\n\n1. insert_end \n 2.Display\n 3.exit");
                printf("\n\n enter your choice ");
                scanf("%d",&ch);

                  switch(ch)
                {
                        case 1:insert_end();
                        display();
                        break;
                        case 2:display();
                        break;
                         case 3: exit(0);
                        break;
                        default:printf("\nwrong coice!");
                        break;
                }
        }
}

void insert_end() {
        int val;
        struct node *new_node,*ptr;
        new_node=(struct node*)(malloc(sizeof(struct node)));
        printf("Enter an element:");
        scanf("%d",&val);
        new_node->data=val;
        if(start==NULL)        //If list is empty
           {
                start=new_node;
           }
           else
           {
                ptr=start;
                while(ptr->next!=start)
        {
                ptr=ptr->next;
        }
                ptr->next=new_node;

           }
        new_node->next=start;
}
void display()
{
        struct node *ptr;
        ptr=start;
        while(ptr->next!=start)
        {
                printf("\nelement is %d",ptr->data);
                ptr=ptr->next;
        }
        printf("\nelement is %d",ptr->data);
}
```

OUTPUT:



```
 ***CIRCULAR LINKLIST MENU***

1. insert_end
 2.Display
 3.exit

 enter your choice 1
Enter an element:98

element is 98
 ***CIRCULAR LINKLIST MENU***

1. insert_end
 2.Display
 3.exit

 enter your choice 1
Enter an element:10

element is 98
element is 10
 ***CIRCULAR LINKLIST MENU***

1. insert_end
 2.Display
 3.exit

 enter your choice 1
Enter an element:777

element is 98
element is 10
element is 777
 ***CIRCULAR LINKLIST MENU***

1. insert_end
 2.Display
 3.exit

 enter your choice
```

Practical – 20

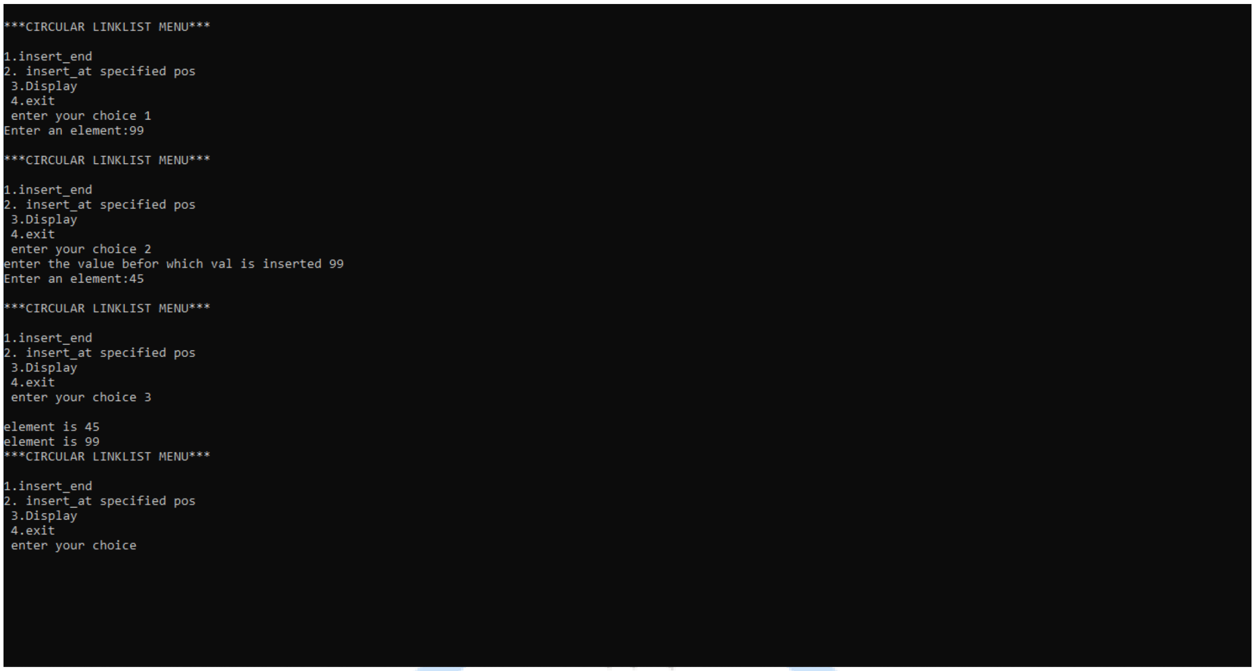
AIM: Write a program to implement the following operation on circular linked list

a) Insert node at specified position.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void insert_beg();
void insert_befpos();
void insert_end();
void display();
struct node {
int data;
struct node *next;
}*start=NULL;

int main() {
    int ch;
    while(1)
    {
        printf("***CIRCULAR LINKLIST MENU***");
        printf("\n\n1.insert_end\n2. insert_at specified pos \n 3.Display\n 4.exit");
        printf("\n enter your choice ");
        scanf("%d",&ch);
        switch(ch)
        {
            //case 1:insert_beg();
            //break;
            case 1:insert_end();
            break;
            case 2:insert_befpos();
            break;
            break;
            case 3:display();
            break;
            case 4: exit(0);
            break;
            default:printf("\nwrong coice!");
            break;
        }
    }
    getch();
}
void insert_beg() {
    struct node *new_node,*ptr;
    int val;
    new_node=(struct node*)(malloc(sizeof(struct node)));
    printf("Enter an element:");
    scanf("%d",&val);
    new_node->data=val;
    ptr=start;
    while(ptr->next!=start)
    {
        ptr=ptr->next;
    }
    new_node->next=start;
    ptr->next=new_node;
    start=new_node;
}
void insert_befpos(){
    struct node *new_node,*ptr,*preptr;
    int val,num;
    new_node=(struct node*)(malloc(sizeof(struct node)));
    printf("enter the value befor which val is inserted");
    scanf("%d",&num);
    if(start->data == num)
    {
        insert_beg();
    }
    else{
        printf("Enter an element:");
        scanf("%d",&val);
        new_node->data=val;
        ptr=start;
        while(ptr->data!=num)
        {
            preptr=ptr;
            ptr=ptr->next;
        }
        new_node->next=ptr;
        preptr->next=new_node;
    }
}
void insert_end() {
    int val;
    struct node *new_node,*ptr;
    new_node=(struct node*)(malloc(sizeof(struct node)));
    printf("Enter an element:");
    scanf("%d",&val);
    new_node->data=val;
    if(start==NULL) //If list is empty
    {
        start=new_node;
    }
    else
    {
        ptr=start;
    while(ptr->next!=start)
    {
        ptr=ptr->next;
    }
    ptr->next=new_node;
    }
    new_node->next=start;
}
void display()
{
    struct node *ptr;
    ptr=start;
    while(ptr->next!=start)
    {
        printf("\nelement is %d",ptr->data);
        ptr=ptr->next;
    }
    printf("\nelement is %d",ptr->data);
}
```

OUTPUT:



```
***CIRCULAR LINKLIST MENU***

1.insert_end
2. insert_at specified pos
 3.Display
 4.exit
 enter your choice 1
Enter an element:99

***CIRCULAR LINKLIST MENU***

1.insert_end
2. insert_at specified pos
 3.Display
 4.exit
 enter your choice 2
enter the value befor which val is inserted 99
Enter an element:45

***CIRCULAR LINKLIST MENU***

1.insert_end
2. insert_at specified pos
 3.Display
 4.exit
 enter your choice 3

element is 45
element is 99
***CIRCULAR LINKLIST MENU***

1.insert_end
2. insert_at specified pos
 3.Display
 4.exit
 enter your choice
```

Practical – 21

AIM: Write a program to implement the following operation on circular linked list
a) Delete the first node

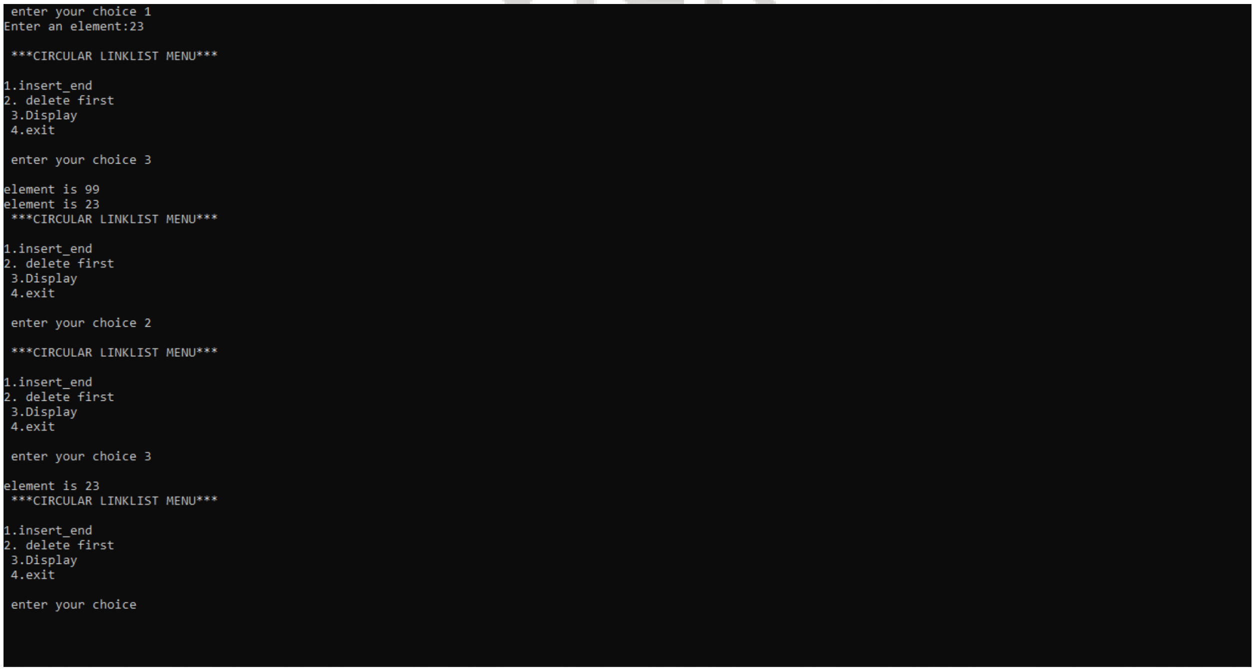
PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void delete_first();
void insert_end();
void display();
struct node {
        int data;
        struct node *next;
}*start=NULL;

int main() {
        int ch;
        while(1)
        {
                printf("\n ***CIRCULAR LINKLIST MENU***");
                printf("\n\n1.insert_end\n2. delete first \n 3.Display\n 4.exit");
                printf("\n\n enter your choice ");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:insert_end();
                        break;
                        case 2:delete_first();
                        break;
                        case 3:display();
                        break;
                        case 4: exit(0);
                        break;
                        default:printf("\nwrong coice!");
                        break;
                }
```

```
    }
    getch();
}
void insert_end() {
    int val;
    struct node *new_node,*ptr;
    new_node=(struct node*)(malloc(sizeof(struct node)));
    printf("Enter an element:");
    scanf("%d",&val);
    new_node->data=val;
    if(start==NULL) //If list is empty
    {
        start=new_node;
    }
    else
    {
        ptr=start;
        while(ptr->next!=start)
            {
                ptr=ptr->next;
            }
        ptr->next=new_node;
    }
    new_node->next=start;
}
void display()
{
    struct node *ptr;
    ptr=start;
    while(ptr->next!=start)
    {
        printf("\nelement is %d",ptr->data);
        ptr=ptr->next;
    }
    printf("\nelement is %d",ptr->data);
}
void delete_first()
{
    struct node *prev=start,*first=start;
    if(start == NULL)
```

```
        {
                printf("list empty");
        }
        else if(prev->next == prev)
        {
                start=NULL;
        }
        else{
                while(prev->next != start)
                {
                        prev=prev->next;
                }
                prev->next = first->next;
                start=prev->next;
                free(first);
        }
}
```

OUTPUT:



```
 enter your choice 1
Enter an element:23

 ***CIRCULAR LINKLIST MENU***

1.insert_end
2. delete first
 3.Display
 4.exit

 enter your choice 3

element is 99
element is 23
 ***CIRCULAR LINKLIST MENU***

1.insert_end
2. delete first
 3.Display
 4.exit

 enter your choice 2

 ***CIRCULAR LINKLIST MENU***

1.insert_end
2. delete first
 3.Display
 4.exit

 enter your choice 3

element is 23
 ***CIRCULAR LINKLIST MENU***

1.insert_end
2. delete first
 3.Display
 4.exit

 enter your choice
```

Practical – 22

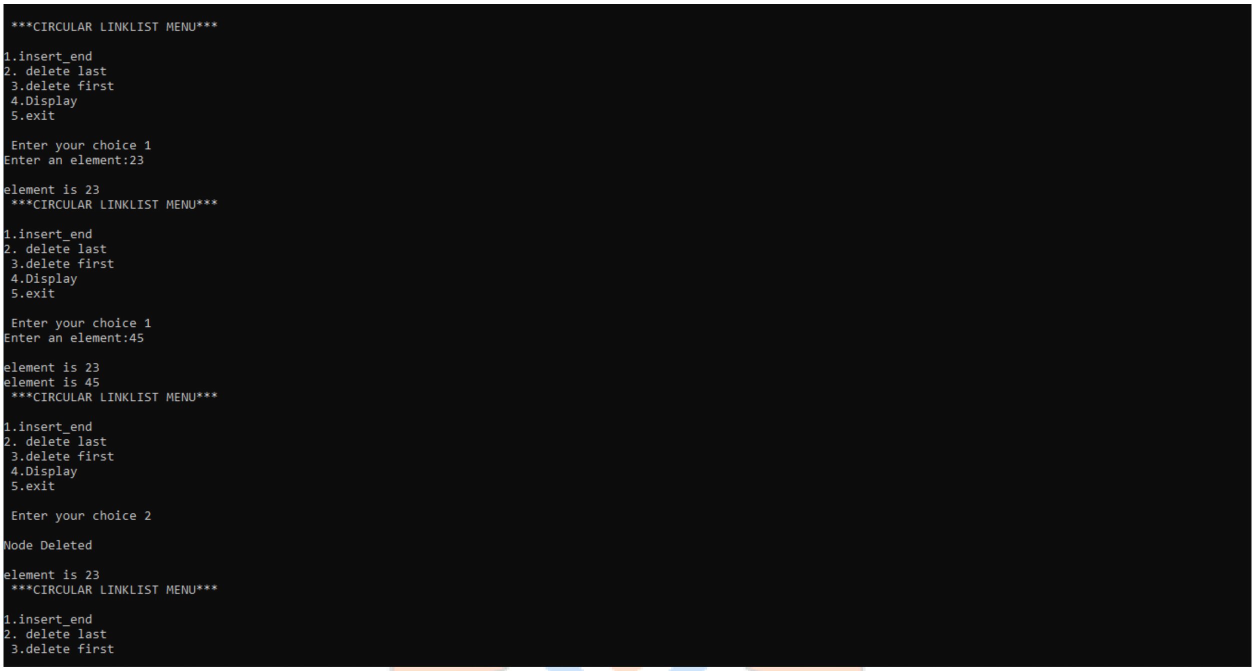
AIM: Write a program to implement the following operation on circular linked list

a) Delete the last node

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void delete_last();
void insert_end();
void delete_first();
void display();

struct node {
int data;
struct node *next;
}*start=NULL;

int main(){
        int ch;
        while(1){
        printf("\n ***CIRCULAR LINKLIST MENU***");
        printf("\n\n1.insert_end\n2. delete last\n 3.delete first \n 4.Display\n 5.exit");
        printf("\n\n Enter your choice ");
        scanf("%d",&ch);
                switch(ch)
                {
                case 1:insert_end();
                break;
                case 2:delete_last();
                break;
                case 3:delete_first();
                break;
                case 4:display();
                break;
                case 5: exit(0);
                break;
                default:printf("\nwrong coice!");
                break;
                }
        }
}
void insert_end() {
        int val;
        struct node *new_node,*ptr;
        new_node=(struct node*)(malloc(sizeof(struct node)));
        printf("Enter an element:");
        scanf("%d",&val);
        new_node->data=val;
                if(start==NULL) //If list is empty
                {
                start=new_node;
                }
                else
                {
                ptr=start;
                while(ptr->next!=start)
                {
                ptr=ptr->next;
                }
                ptr->next=new_node;
                }
                new_node->next=start;
                display();
        }
void display(){
        struct node *ptr;
        ptr=start;
        while(ptr->next!=start)
        {
                printf("\nelement is %d",ptr->data);
                ptr=ptr->next;
        }
        printf("\nelement is %d",ptr->data);
}
void delete_last(){
        struct node *ptr, *preptr;
                if(start==NULL)
                {
                printf("\nUNDERFLOW\n");
                }
                else if (start ->next == start)
                {
                start= NULL;
                free(start);
                printf("\nNode Deleted\n");
                }
                else
                {
                ptr = start;
                while(ptr ->next != start)
                {
                preptr=ptr;
                ptr = ptr->next;
                }
        preptr->next = ptr -> next;
        free(ptr);
        printf("\nNode Deleted\n");
        display();
        }
}

void delete_first(){
        struct node *prev=start,*first=start;
                if(start == NULL)
                {
                    printf("list empty");
                }
                else if(prev->next == prev)
                {
                    start=NULL;
                }
                else{
                    while(prev->next != start)
                    {
                    prev=prev->next;
                    }
        prev->next = first->next;
        start=prev->next;
        free(first);
        display();
        }
}
```

OUTPUT:



```
 ***CIRCULAR LINKLIST MENU***

1.insert_end
2. delete last
 3.delete first
 4.Display
 5.exit

 Enter your choice 1
Enter an element:23

element is 23
 ***CIRCULAR LINKLIST MENU***

1.insert_end
2. delete last
 3.delete first
 4.Display
 5.exit

 Enter your choice 1
Enter an element:45

element is 23
element is 45
 ***CIRCULAR LINKLIST MENU***

1.insert_end
2. delete last
 3.delete first
 4.Display
 5.exit

 Enter your choice 2

Node Deleted

element is 23
 ***CIRCULAR LINKLIST MENU***

1.insert_end
2. delete last
 3.delete first
```

Practical – 23

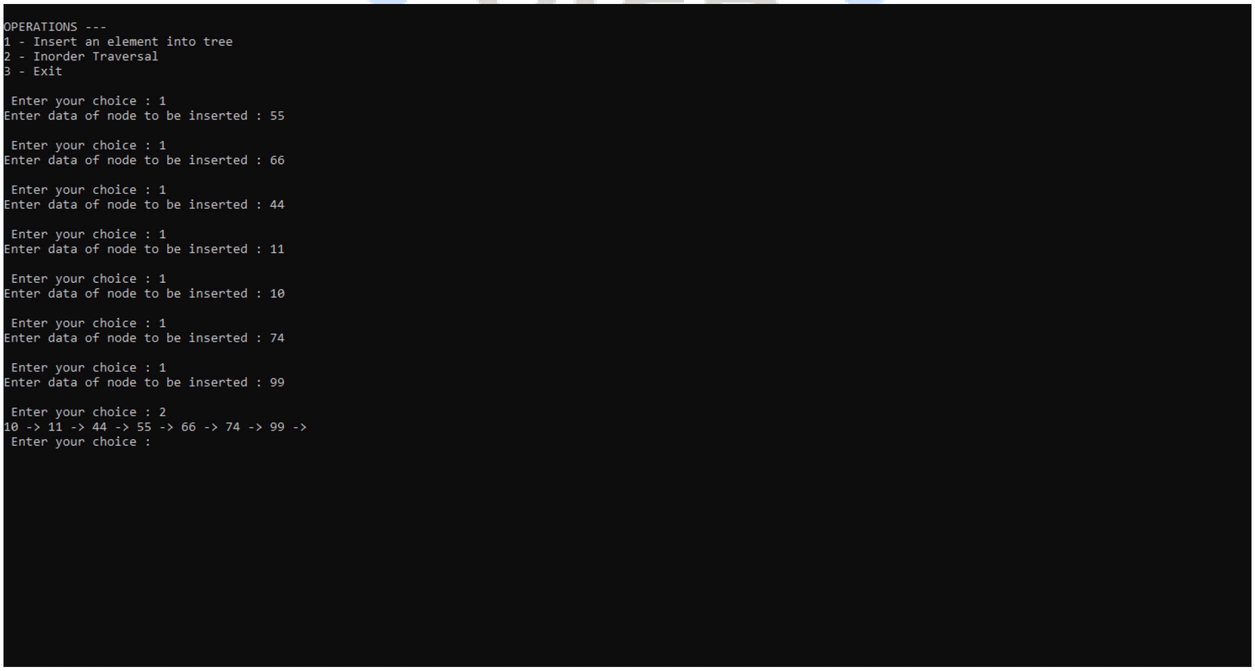
AIM: Implement recursive or non-recursive tree traversing methods of inorder traversal.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
struct btnode
{
        int value;
        struct btnode *l;
        struct btnode *r;
}*root = NULL, *temp = NULL, *t2, *t1;

void insert();
void inorder(struct btnode *t);
void create();
void search(struct btnode *t);

int main()
{
        int ch;
        printf("\nOPERATIONS ---");
        printf("\n1 - Insert an element into tree\n");
        printf("2 - Inorder Traversal\n");
        printf("3 - Exit\n");
        while(1)
        {
                printf("\n Enter your choice : ");
                scanf("%d", &ch);
                switch (ch)
                {
                        case 1:
                        insert();
                        break;
                        case 2:
                        inorder(root);
                        break;
                        case 3:
                        exit(0);
                        default :
                        printf("Wrong choice, Please enter correct choice ");
                        break;
                }
        }
}
/* To insert a node in the tree */
void insert()
{
        create();
        if (root == NULL)
        root = temp;
        else
        search(root);
}
/* To create a node */
void create()
{
        int data;
        printf("Enter data of node to be inserted : ");
        scanf("%d", &data);
        temp = (struct btnode *)malloc(sizeof(struct btnode));
        temp->value = data;
        temp->l = temp->r = NULL;
}
/* Function to search the appropriate position to insert the new node */
void search(struct btnode *t)
{
        if ((temp->value > t->value) && (t->r != NULL)) /* value more than root node value
insert at right */
                search(t->r);
        else if ((temp->value > t->value) && (t->r == NULL))
                t->r = temp;
        else if ((temp->value < t->value) && (t->l != NULL)) /* value less than root node value
insert at left */
                search(t->l);
        else if ((temp->value < t->value) && (t->l == NULL))
```

```
                t->l = temp;
}
/* recursive function to perform inorder traversal of tree */
void inorder(struct btnode *t)
{
        if (root == NULL)
        {
                printf("No elements in a tree to display");
                return;
        }
        if (t->l != NULL)
                inorder(t->l);
        printf("%d -> ", t->value);
        if (t->r != NULL)
                inorder(t->r);
}
```

OUTPUT:



```
OPERATIONS ---
1 - Insert an element into tree
2 - Inorder Traversal
3 - Exit

 Enter your choice : 1
Enter data of node to be inserted : 55

 Enter your choice : 1
Enter data of node to be inserted : 66

 Enter your choice : 1
Enter data of node to be inserted : 44

 Enter your choice : 1
Enter data of node to be inserted : 11

 Enter your choice : 1
Enter data of node to be inserted : 10

 Enter your choice : 1
Enter data of node to be inserted : 74

 Enter your choice : 1
Enter data of node to be inserted : 99

 Enter your choice : 2
10 -> 11 -> 44 -> 55 -> 66 -> 74 -> 99 ->
 Enter your choice :
```

Practical – 24

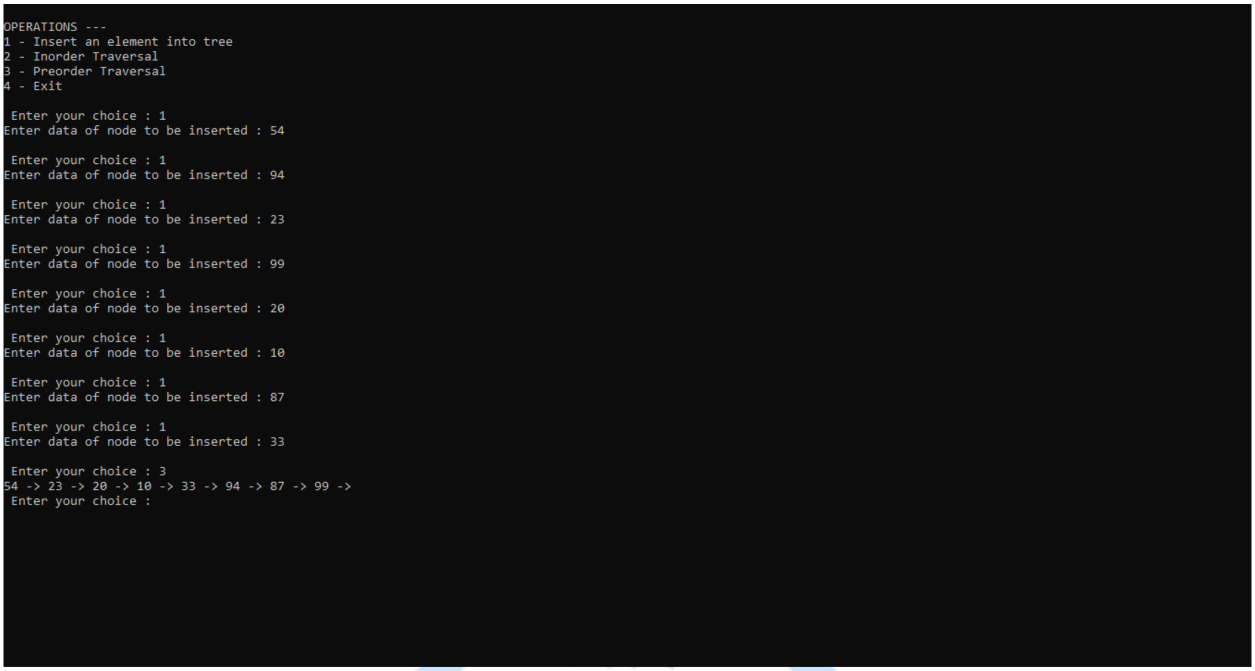
AIM: Implement recursive or non-recursive tree traversing methods of Preorder traversal.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
struct btnode
{
    int value;
    struct btnode *l;
    struct btnode *r;
}*root = NULL, *temp = NULL, *t2, *t1;

void insert();
void inorder(struct btnode *t);
void preorder(struct btnode *t);
void create();
void search(struct btnode *t);

int main()
{
    int ch;
    printf("\nOPERATIONS ---");
    printf("\n1 - Insert an element into tree\n");
    printf("2 - Inorder Traversal\n");
    printf("3 - Preorder Traversal\n");
    printf("4 - Exit\n");
    while(1)
    {
        printf("\n Enter your choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1:
            insert();
            break;
            case 2:
            inorder(root);
            break;
            case 3:
            preorder(root);
            break;
            case 4:
            exit(0);
            default :
            printf("Wrong choice, Please enter correct choice ");
            break;
        }
    }
}
/* To insert a node in the tree */
void insert()
{
    create();
    if (root == NULL)
    root = temp;
    else
    search(root);
}
/* To create a node */
void create()
{
    int data;
    printf("Enter data of node to be inserted : ");
    scanf("%d", &data);
    temp = (struct btnode *)malloc(sizeof(struct btnode));
    temp->value = data;
    temp->l = temp->r = NULL;
}
/* Function to search the appropriate position to insert the new node */
void search(struct btnode *t)
{
    if ((temp->value > t->value) && (t->r != NULL)) /* value more than root node value
insert at right */
    search(t->r);
    else if ((temp->value > t->value) && (t->r == NULL))
```

```
        t->r = temp;
        else if ((temp->value < t->value) && (t->l != NULL)) /* value less than root node value
insert at left */
        search(t->l);
        else if ((temp->value < t->value) && (t->l == NULL))
        t->l = temp;
}
/* recursive function to perform inorder traversal of tree */
void inorder(struct btnode *t)
{
        if (root == NULL)
        {
        printf("No elements in a tree to display");
        return;
        }
        if (t->l != NULL)
        inorder(t->l);
        printf("%d -> ", t->value);
        if (t->r != NULL)
        inorder(t->r);
}


void preorder(struct btnode *t)
{
if (root == NULL)
        {
        printf("No elements in a tree to display");
        return;
        }
        printf("%d -> ", t->value);
        if (t->l != NULL)
        preorder(t->l);
        if (t->r != NULL)
        preorder(t->r);
}
```

OUTPUT:



```
OPERATIONS ---
1 - Insert an element into tree
2 - Inorder Traversal
3 - Preorder Traversal
4 - Exit

 Enter your choice : 1
Enter data of node to be inserted : 54

 Enter your choice : 1
Enter data of node to be inserted : 94

 Enter your choice : 1
Enter data of node to be inserted : 23

 Enter your choice : 1
Enter data of node to be inserted : 99

 Enter your choice : 1
Enter data of node to be inserted : 20

 Enter your choice : 1
Enter data of node to be inserted : 10

 Enter your choice : 1
Enter data of node to be inserted : 87

 Enter your choice : 1
Enter data of node to be inserted : 33

 Enter your choice : 3
54 -> 23 -> 20 -> 10 -> 33 -> 94 -> 87 -> 99 ->
 Enter your choice :
```

Practical – 25

AIM: Implement recursive or non-recursive tree traversing methods of postorder traversal.

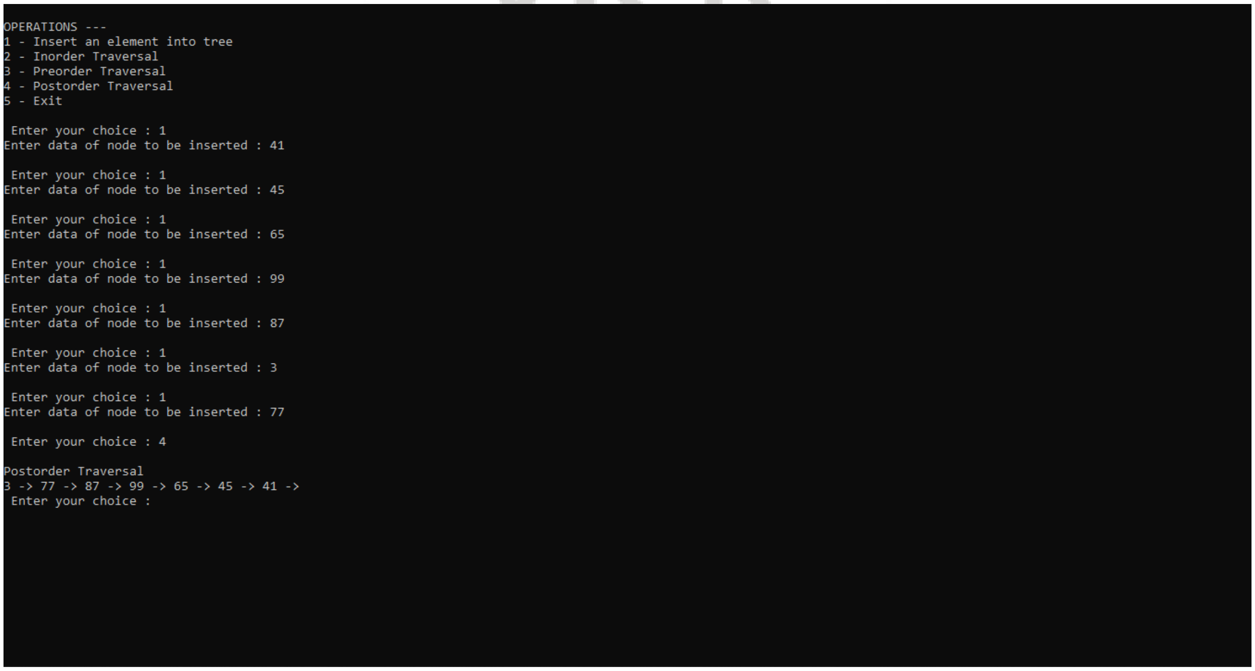
PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
struct btnode
{
        int value;
        struct btnode *l;
        struct btnode *r;
}*root = NULL, *temp = NULL, *t2, *t1;

void insert();
void inorder(struct btnode *t);
void preorder(struct btnode *t);
void postorder(struct btnode *t);
void create();
void search(struct btnode *t);

int main()
{
        int ch;
        printf("\nOPERATIONS ---");
        printf("\n1 - Insert an element into tree\n");
        printf("2 - Inorder Traversal\n");
        printf("3 - Preorder Traversal\n");
        printf("4 - Postorder Traversal\n");
        printf("5 - Exit\n");
        while(1)
        {
                printf("\n Enter your choice : ");
                scanf("%d", &ch);
                switch (ch)
                {
                        case 1:
```

```
            insert();
            break;
            case 2:
                printf("\nInorder Traversal\n");
            inorder(root);
            break;
            case 3:
                printf("\nPreorder Traversal\n");
            preorder(root);
            break;
            case 4:
                printf("\nPostorder Traversal\n");
            postorder(root);
            break;
            case 5:
            exit(0);
            default :
            printf("Wrong choice, Please enter correct choice ");
            break;
        }
    }
}
/* To insert a node in the tree */
void insert()
{
    create();
    if (root == NULL)
        root = temp;
    else
        search(root);
}
/* To create a node */
void create()
{
    int data;
    printf("Enter data of node to be inserted : ");
    scanf("%d", &data);
    temp = (struct btnode *)malloc(sizeof(struct btnode));
    temp->value = data;
    temp->l = temp->r = NULL;
```

```
}
/* Function to search the appropriate position to insert the new node */
void search(struct btnode *t)
{
        if ((temp->value > t->value) && (t->r != NULL)) /* value more than root node value
insert at right */
        search(t->r);
        else if ((temp->value > t->value) && (t->r == NULL))
        t->r = temp;
        else if ((temp->value < t->value) && (t->l != NULL)) /* value less than root node value
insert at left */
        search(t->l);
        else if ((temp->value < t->value) && (t->l == NULL))
        t->l = temp;
}
/* recursive function to perform inorder traversal of tree */
void inorder(struct btnode *t)
{
        if (root == NULL)
        {
        printf("No elements in a tree to display");
        return;
        }
        if (t->l != NULL)
        inorder(t->l);
        printf("%d -> ", t->value);
        if (t->r != NULL)
        inorder(t->r);
}


void preorder(struct btnode *t)
{
        if (root == NULL)
        {
        printf("No elements in a tree to display");
        return;
        }
        printf("%d -> ", t->value);
        if (t->l != NULL)
        preorder(t->l);
        if (t->r != NULL)
        preorder(t->r);
}

void postorder(struct btnode *t){
        if (root == NULL)
        {
                printf("No elements in a tree to display");
                return;
        }
        if (t->l != NULL)
                postorder(t->l);
        if (t->r != NULL)
                postorder(t->r);
        printf("%d -> ", t->value);
}
```

OUTPUT:



```
OPERATIONS ---
1 - Insert an element into tree
2 - Inorder Traversal
3 - Preorder Traversal
4 - Postorder Traversal
5 - Exit

 Enter your choice : 1
Enter data of node to be inserted : 41

 Enter your choice : 1
Enter data of node to be inserted : 45

 Enter your choice : 1
Enter data of node to be inserted : 65

 Enter your choice : 1
Enter data of node to be inserted : 99

 Enter your choice : 1
Enter data of node to be inserted : 87

 Enter your choice : 1
Enter data of node to be inserted : 3

 Enter your choice : 1
Enter data of node to be inserted : 77

 Enter your choice : 4

Postorder Traversal
3 -> 77 -> 87 -> 99 -> 65 -> 45 -> 41 ->
 Enter your choice :
```

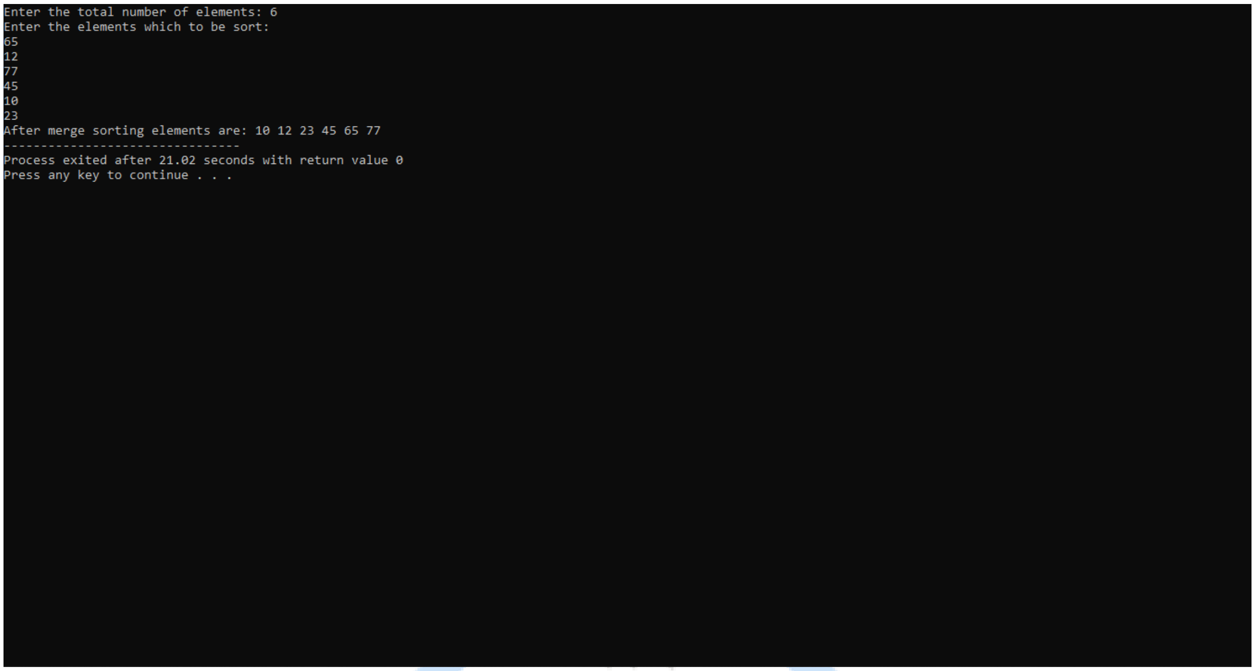
Practical – 26

AIM: Write a program to implement Merge Sort

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#define MAX 50

void mergeSort(int arr[],int low,int mid,int high);
void partition(int arr[],int low,int high);

int main(){
    int merge[MAX],i,n;
        printf("YASH PATIL 19CE032\n");
        printf("Enter the total number of elements: ");
        scanf("%d",&n);
        printf("Enter the elements which to be sort: \n");
            for(i=0;i<n;i++){
                scanf("%d",&merge[i]);
            }
    partition(merge,0,n-1);
        printf("After merge sorting elements are: ");
            for(i=0;i<n;i++){
                printf("%d ",merge[i]);
            }
    return 0;
}
void partition(int arr[],int low,int high){
    int mid;
        if(low<high){
        mid=(low+high)/2;
        partition(arr,low,mid);
        partition(arr,mid+1,high);
        mergeSort(arr,low,mid,high);
    }
}
void mergeSort(int arr[],int low,int mid,int high){
    int i,m,k,l,temp[MAX];
    l=low;
    i=low;
    m=mid+1;
    while((l<=mid)&&(m<=high)){
    if(arr[l]<=arr[m]){
        temp[i]=arr[l];
        l++;
    }
    else{
        temp[i]=arr[m];
        m++;
    }
    i++;
    }
    if(l>mid){
        for(k=m;k<=high;k++){
        temp[i]=arr[k];
        i++;
        }
    }
    else{
        for(k=l;k<=mid;k++)
        {
        temp[i]=arr[k];
        i++;
        }
    }
    for(k=low;k<=high;k++)
    {
        arr[k]=temp[k];
    }
}
```

OUTPUT:



```
Enter the total number of elements: 6
Enter the elements which to be sort:
65
12
77
45
10
23
After merge sorting elements are: 10 12 23 45 65 77
--------------------------------
Process exited after 21.02 seconds with return value 0
Press any key to continue . . .
```

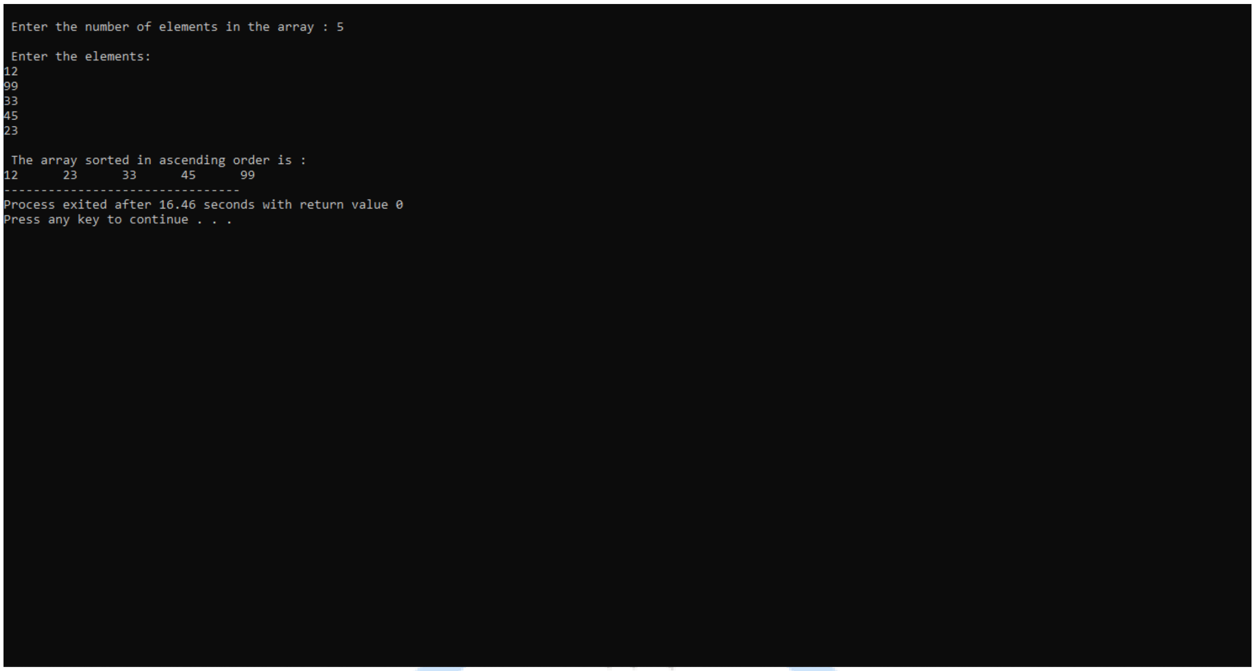
Practical – 27

AIM: Write a program to implement Bubble Sort

PROGRAM:

```
#include <stdio.h>

int main()
{
int i, n, temp, j, arr[10];

printf("\n Enter the number of elements in the array : ");
scanf("%d", &n);
printf("\n Enter the elements: \n");
for(i=0;i<n;i++)
{
scanf("%d", &arr [i]);
}
for(i=0;i<n;i++){
        for(j=0;j<n-i-1;j++)
        {
                if(arr[j] > arr[j+1])
                {
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
                }
        }
}
printf("\n The array sorted in ascending order is :\n");
for(i=0;i<n;i++)
        printf("%d\t", arr[i]);
//return 0;
}
```

OUTPUT:



```
 Enter the number of elements in the array : 5

 Enter the elements:
12
99
33
45
23

 The array sorted in ascending order is :
12      23      33      45      99
--------------------------------
Process exited after 16.46 seconds with return value 0
Press any key to continue . . .
```

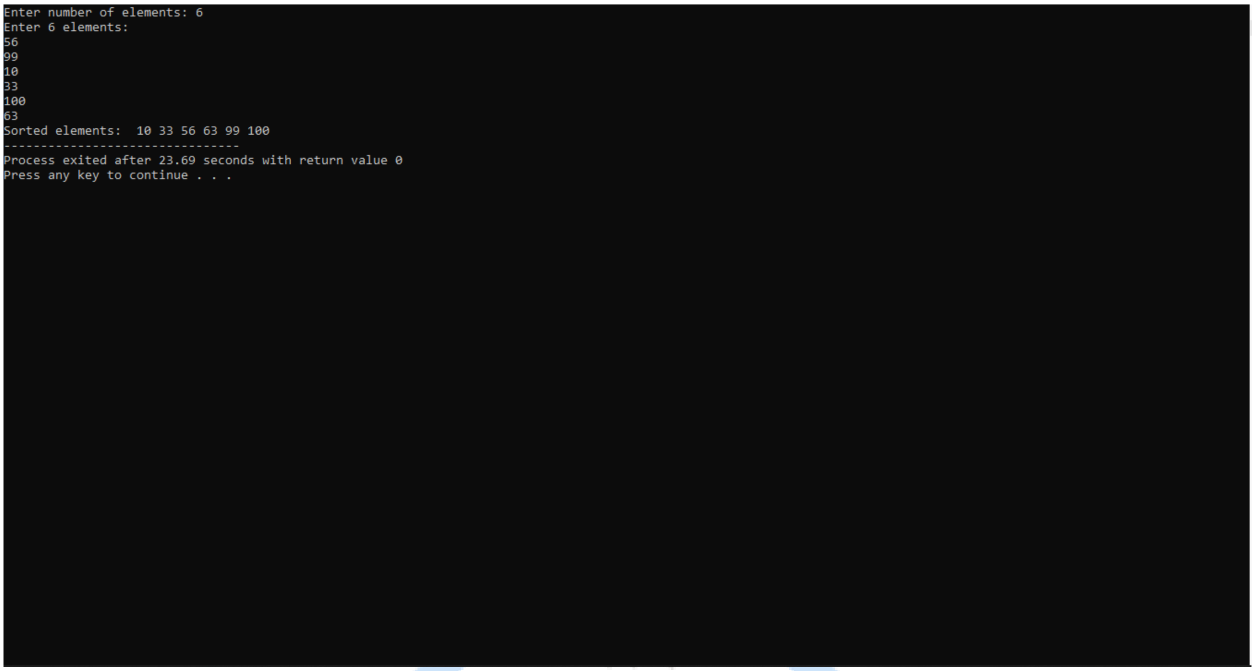
Practical – 28

AIM: Write a program to implement Selection Sort

PROGRAM:

```
#include<stdio.h>

int main(){

  int i, j, count, temp, number[25];

  printf("Enter number of elements: ");
  scanf("%d",&count);

  printf("Enter %d elements: ", count);

  for(i=0;i<count;i++)
    scanf("%d",&number[i]);

  for(i=0;i<count;i++){
    for(j=i+1;j<count;j++){
      if(number[i]>number[j]){
        temp=number[i];
        number[i]=number[j];
        number[j]=temp;
      }
    }
  }

  printf("Sorted elements: ");
  for(i=0;i<count;i++)
    printf(" %d",number[i]);

  return 0;
}
```

OUTPUT:



```
Enter number of elements: 6
Enter 6 elements:
56
99
10
33
100
63
Sorted elements:  10 33 56 63 99 100
--------------------------------
Process exited after 23.69 seconds with return value 0
Press any key to continue . . .
```