VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
on

Data Structures (21CS3PCDST)

Submitted by

Sannidhi M (1BM21CS189)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
October-2022 to Feb-2023

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled “Data Structures (21CS3PCDST)” carried out by **Sannidhi M (1BM21CS189),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a Data Structures (21CS3PCDST) work prescribed for the said degree.

Dr. Pallavi G B
Associate Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

INDEX

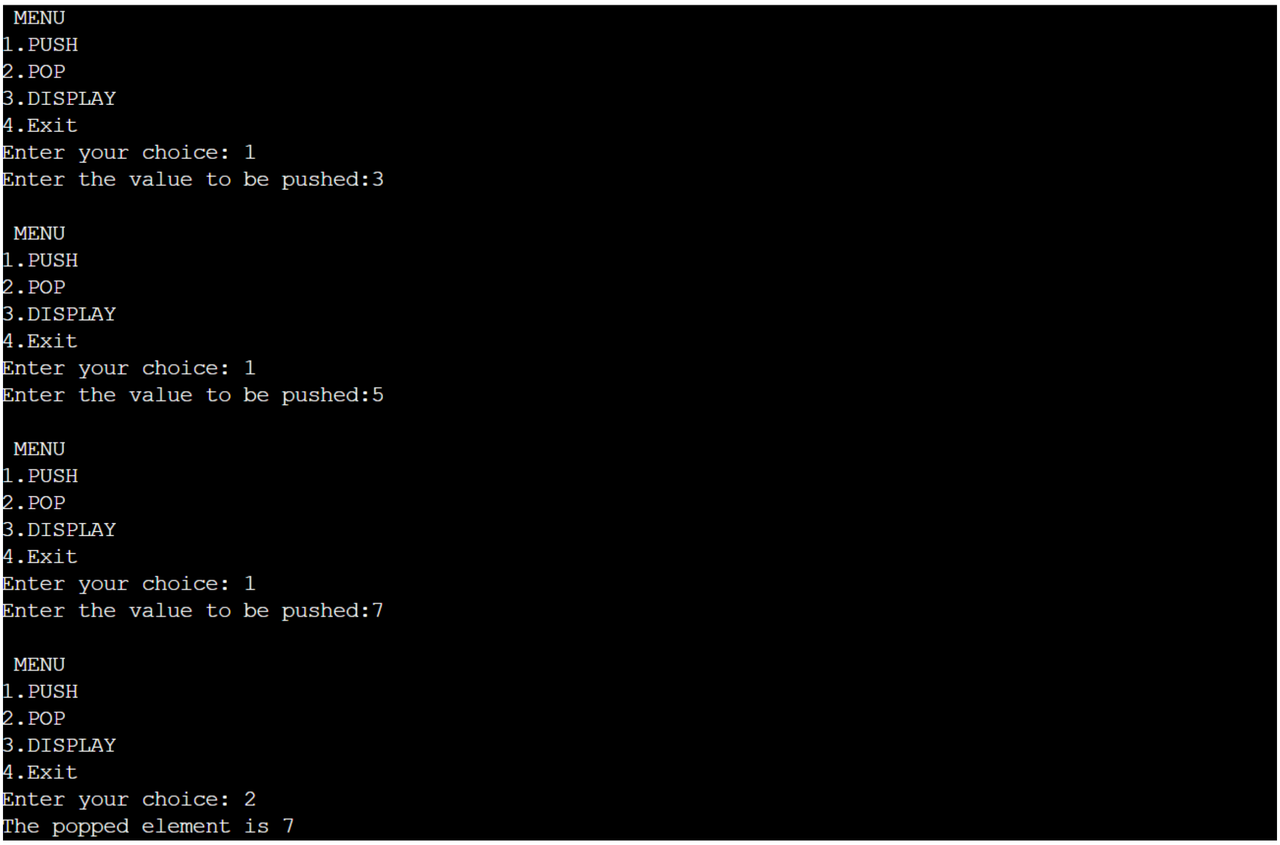
PROGRAM 1

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, stack underflow

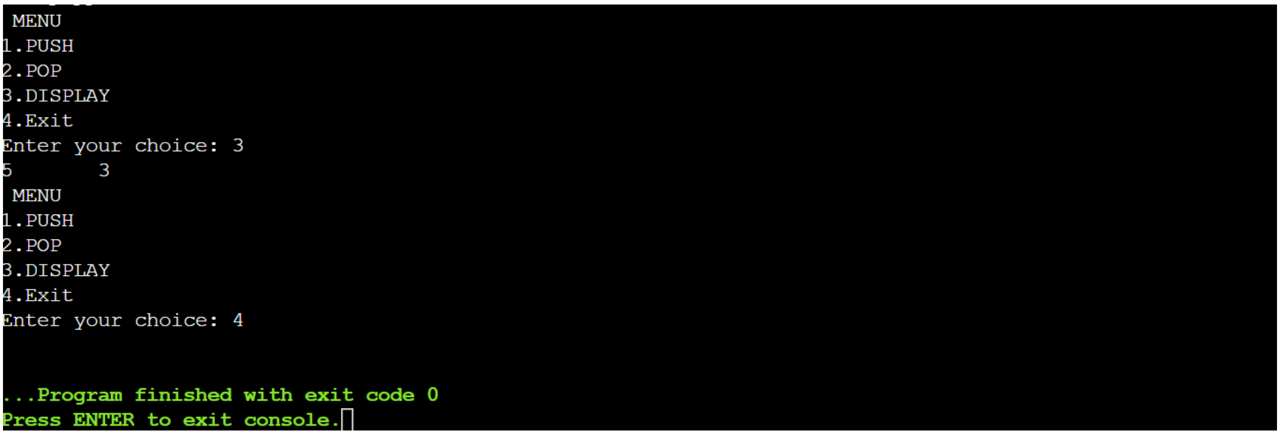
CODE

```
#include<stdio.h>
#include<conio.h>
int st[100];
int n=101,top=-1,x;
void push();
void pop();
void display();
int main()
{
    int choice,value;
    do
    {
     printf("\n MENU");
     printf("\n1.PUSH \n2.POP \n3.DISPLAY \n4.Exit");
     printf("\nEnter your choice: ");
     scanf("%d",&choice);
  switch(choice)
  {
   case 1:push();
          break;

   case 2:pop();
          break;

   case 3:display();
          break;

   case 4: break;

   default: printf("Enter valid choice");
    }
}
while(choice!=4);
return 0;
}
void push()
{
    if(top>n-1)
    printf("Stack Overflow");
    else
    {
        printf("Enter the value to be pushed:");
        scanf("%d",&x);
        top++;
        st[top]=x;
    }
}
void pop()
{
    if(top<=-1)
    printf("Stack Underflow");
    else
    {
      x=st[top];
      printf("The popped element is %d",x);
      top--;
    }
}
void display()
{
    int i;
    for(i=top;i>=0;i--)
    printf("%d \t",st[i]);
}
```

OUTPUT



```
 MENU
1.PUSH
2.POP
3.DISPLAY
4.Exit
Enter your choice: 1
Enter the value to be pushed:3

 MENU
1.PUSH
2.POP
3.DISPLAY
4.Exit
Enter your choice: 1
Enter the value to be pushed:5

 MENU
1.PUSH
2.POP
3.DISPLAY
4.Exit
Enter your choice: 1
Enter the value to be pushed:7

 MENU
1.PUSH
2.POP
3.DISPLAY
4.Exit
Enter your choice: 2
The popped element is 7
```



```
 MENU
1.PUSH
2.POP
3.DISPLAY
4.Exit
Enter your choice: 3
5       3
 MENU
1.PUSH
2.POP
3.DISPLAY
4.Exit
Enter your choice: 4


...Program finished with exit code 0
Press ENTER to exit console.
```

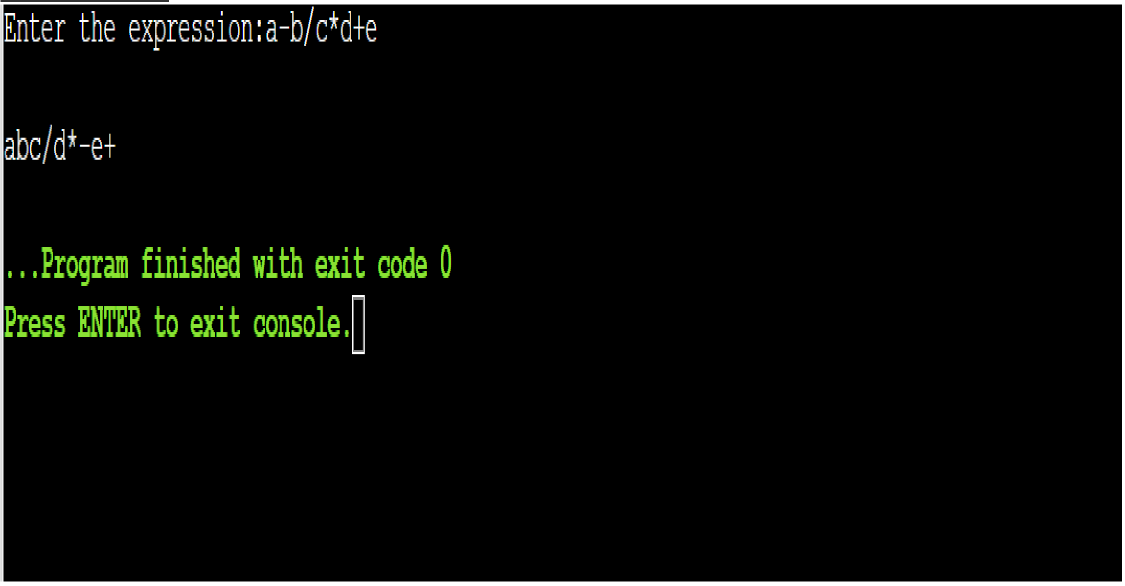
PROGRAM 2

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)

CODE

```
#include<stdio.h>
#include<ctype.h>
char stack[100];
int top =-1;
void push(char x)
{
    stack[++top]=x;
}
char pop()
{
    if (top==-1)
        return -1;
    else
        return stack[top--];
}
int priority (char x)
{

    if(x=='(')
        return 0;
    if(x=='+'|| x=='-')
        return 1;
    if (x=='*' || x=='/')
        return 2;
    return 0;
}
int main()
{
    char exp[100];
    char *e,x;
    printf("Enter the expression:");
    scanf("%s", exp);
    printf("\n");
    e = exp;
    while(*e != '\0')
    {
        if(isalnum(*e))
            printf("%c",*e);
        else if(*e=='(')
                push(*e);
        else if(*e==')')
        {
            while((x=pop())!='(')
                printf("%c",x);
        }
        else
        {
            while(priority(stack[top])>=priority(*e))
                printf("%c",pop());
            push(*e);
        }
        e++;
    }
    while(top!=-1)
    {

        printf("%c",pop());
    }
    return 0;
}
```

OUTPUT



```
Enter the expression:a-b/c*d+e


abc/d*-e+


...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM 3

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

b) Delete

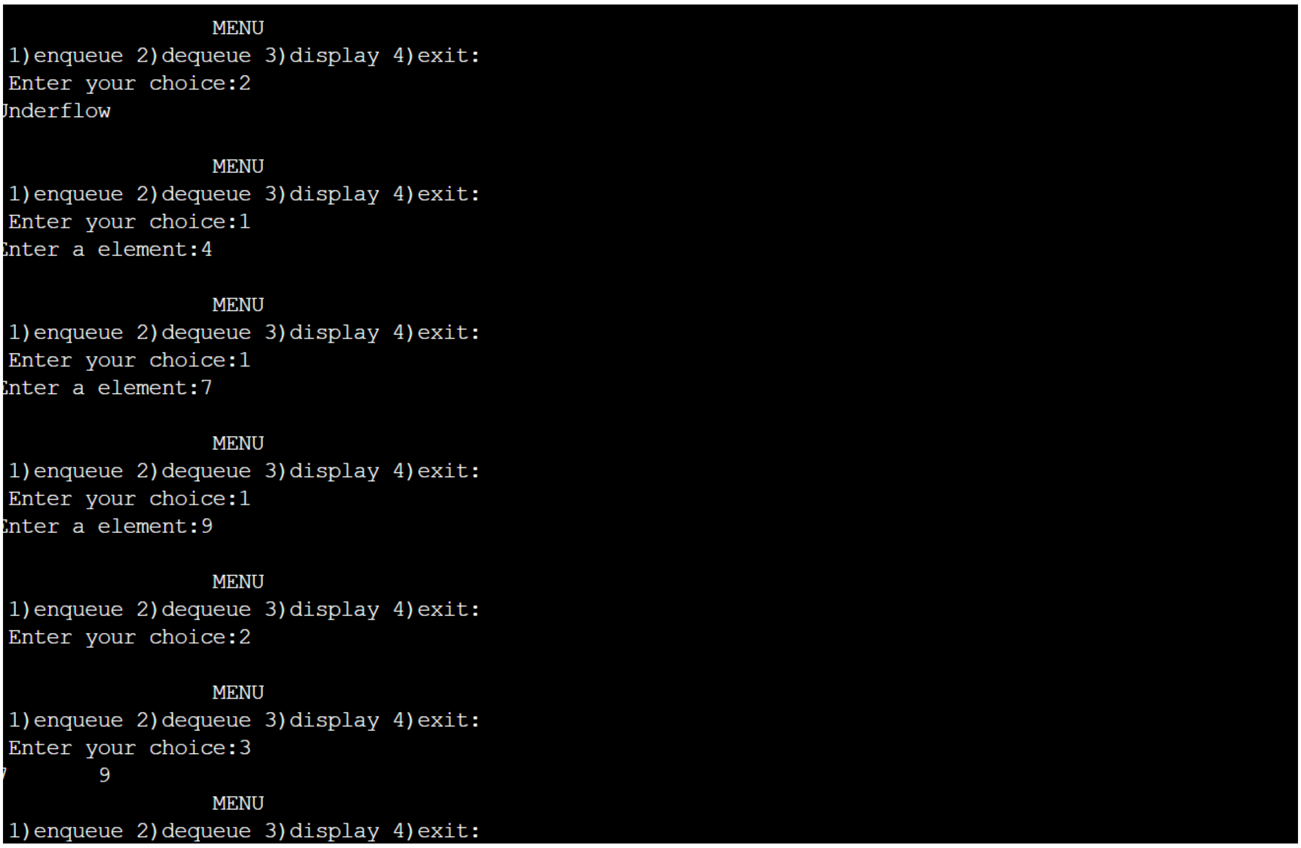
c) Display The program should print appropriate messages for queue empty and queue overflow conditions

CODE

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define MAX 100

int q[MAX];
int rear = -1;
int front = -1;

void enqueue();
void dequeue();
void display();

int main()
{
   int choice;
   do{

   printf("\n\t\t MENU\t\t\n");
   printf(" 1)enqueue 2)dequeue 3)display 4)exit:");
   printf("\n Enter your choice:");
   scanf("%d",&choice);

      switch(choice)
      {
      case 1:
         enqueue();
         break;
      case 2:
         dequeue();
         break;
      case 3:
         display();
         break;
      case 4:
         exit(0);

      default:
         printf("Enter a valid choice");
      }
   } while(choice!=4);
   return 0;
}

void enqueue()
{
   int num;
   if(rear==MAX-1)
   {
      printf("Queue overflow");
   }
   else if((front==-1) && (rear==-1))
   {
      front=0;
      rear=0;
      printf("Enter a element:");
      scanf("%d",&num);
      q[rear]=num;
      rear++;
   }
   else
   {

      printf("Enter a element:");
      scanf("%d",&num);
      q[rear]=num;
      rear++;
   }


}

void dequeue()
{
   int num1;
   if((front==-1)||(front>rear))
   {
      printf("Underflow\n");
   }
   else
   {
      num1=q[front];
      front++;
   }
}

void display()
{
   if((front==-1 )&& (rear==-1))
   {
      printf("Queue is empty!!!\n");
```

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

OUTPUT



```
                MENU
 1)enqueue 2)dequeue 3)display 4)exit:
 Enter your choice:2
Underflow

                MENU
 1)enqueue 2)dequeue 3)display 4)exit:
 Enter your choice:1
Enter a element:4

                MENU
 1)enqueue 2)dequeue 3)display 4)exit:
 Enter your choice:1
Enter a element:7

                MENU
 1)enqueue 2)dequeue 3)display 4)exit:
 Enter your choice:1
Enter a element:9

                MENU
 1)enqueue 2)dequeue 3)display 4)exit:
 Enter your choice:2

                MENU
 1)enqueue 2)dequeue 3)display 4)exit:
 Enter your choice:3
7       9
                MENU
 1)enqueue 2)dequeue 3)display 4)exit:
```

PROGRAM 4

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

a) Insert

b) Delete

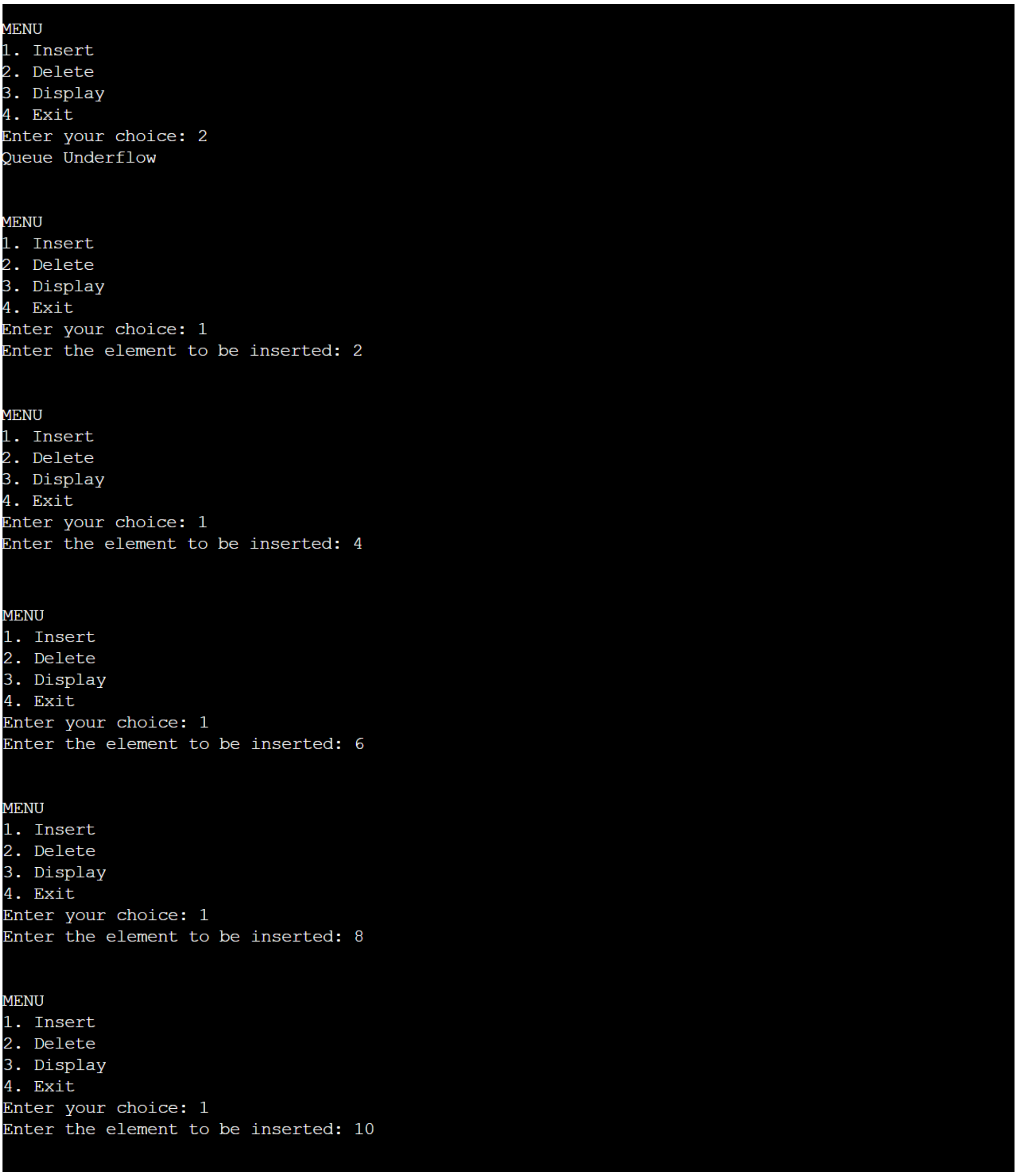
c) Display. The program should print appropriate messages for queue empty and queue overflow conditions.

CODE

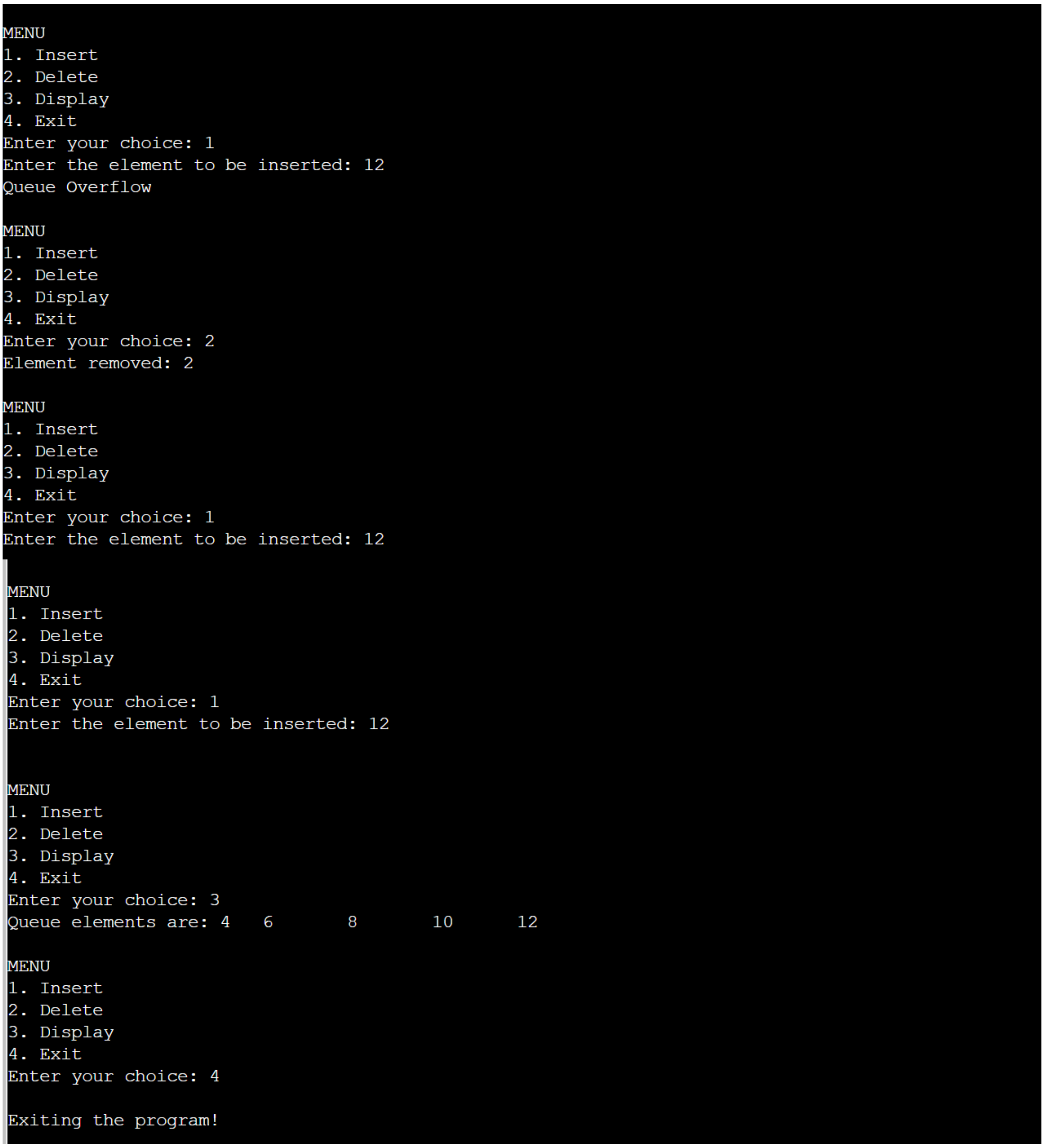
```
#include<stdio.h>
#include<stdlib.h>
int front =-1 , rear=-1 , max=5;
```

```
void insert(int q[])
{
int e;
printf("Enter the element to be inserted: ");
scanf("%d",&e);
if((front==0 && rear==max-1)||(rear==front-1))
{
  printf("Queue Overflow");
}
else if((front==-1)&&(rear==-1))
{
  front=0;
  rear=0;
  q[rear]=e;
}
else if(rear==max-1 && front!=0)
{
  rear=0;
  q[rear]=e;
}
else
{
  rear++;
 q[rear]=e;
}
}

void delete(int q[])
{
int e;
if(rear==-1 && front==-1)
{
printf("Queue Underflow\n");
}
else
{
e=q[front];
printf("Element removed: %d",e);
if(front == rear)
{
front = -1;

rear = -1;
}
else
{
if(front==max-1)
front=0;
else
front++;
}
}
}

void display(int q[])
{
int i;
if(front==-1 && rear==-1)
printf("Queue is empty\n");
else
{
printf("Queue elements are: ");
if(front<rear)
{
for(i=front;i<=rear;i++)
printf("%d\t",q[i]);
}
else
{
for(i=front;i<max;i++)
printf("%d\t",q[i]);
for(i=0;i<=rear;i++)
printf("%d\t",q[i]);
}
}
}

int main()
{
int q[100],choice,e;
do
{
printf("\n\nMENU\n1. Insert\n2. Delete\n3. Display\n4. Exit\nEnter your choice: ");
scanf("%d",&choice);
switch(choice)
{
case 1: insert(q);
break;
case 2: delete(q);
break;
case 3: display(q);
break;
case 4: printf("\nExiting the program!");
exit(0);
break;
default: printf("Invalid Choice!");
break;
}
} while (1);
}
```

OUTPUT



```
MENU
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
Queue Underflow


MENU
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the element to be inserted: 2


MENU
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the element to be inserted: 4


MENU
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the element to be inserted: 6


MENU
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the element to be inserted: 8


MENU
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the element to be inserted: 10
```



```
MENU
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the element to be inserted: 12
Queue Overflow

MENU
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 2
Element removed: 2

MENU
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the element to be inserted: 12

MENU
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter the element to be inserted: 12


MENU
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Queue elements are: 4   6       8       10      12

MENU
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 4

Exiting the program!
```

PROGRAM 5

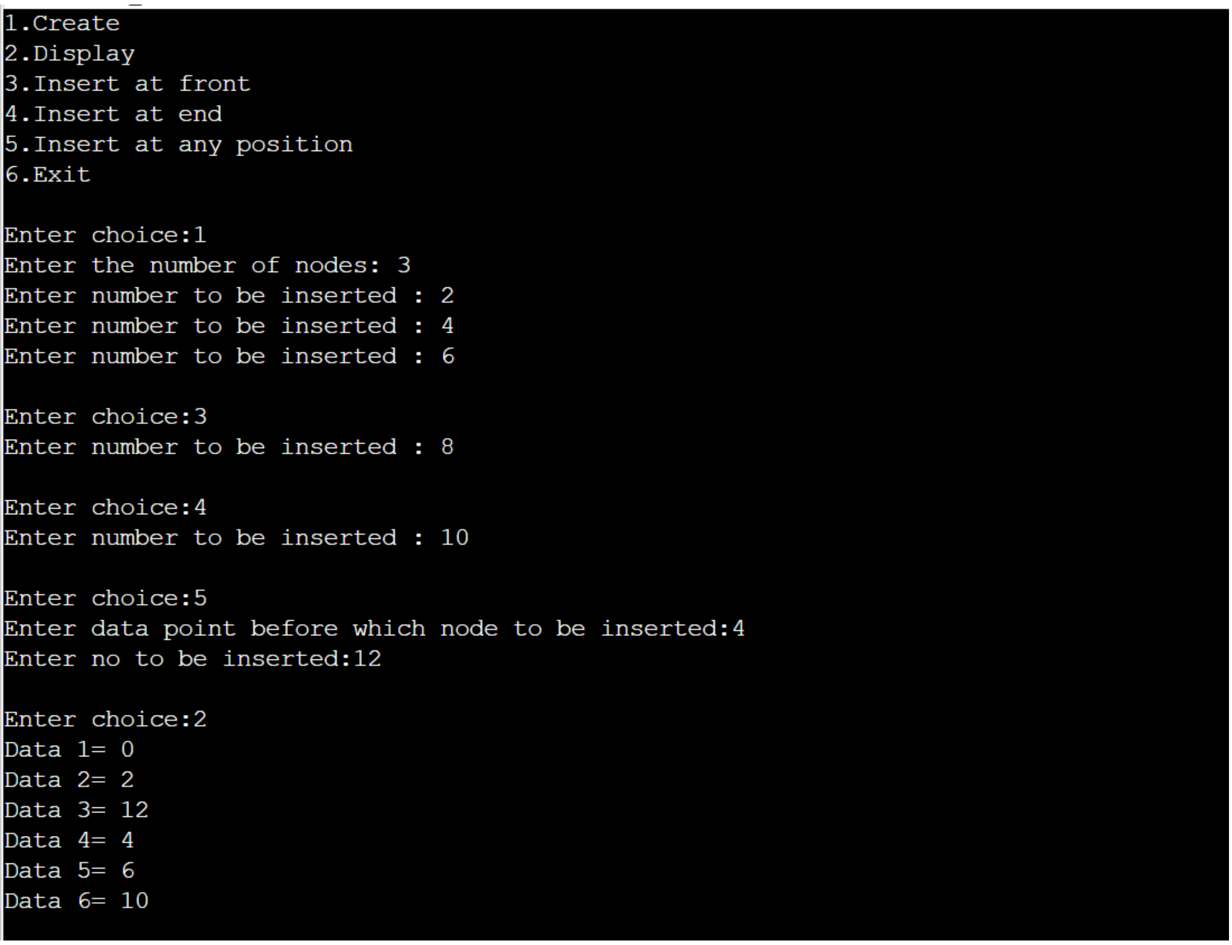
WAP to Implement Singly Linked List with following operations
a) Create a linked list.
b) Insertion of a node at first position, at any position and at end of list.
c) Display the contents of the linked list

CODE

```
#include<stdio.h>
#include<stdlib.h>
struct node {
   int info;
   struct node* next;
```

```
};

struct node* head = NULL;

void createlist()
{
   int n;
   printf("Enter the number of nodes: ");
   scanf("%d", &n);
   int data;
   struct node* newnode;
   struct node* p;
   newnode = malloc(sizeof(struct node));
   head = newnode;
   p = head;
   printf("Enter number to be inserted : ");
   scanf("%d", &data);
   head ->info = data;

   for (int i = 1; i < n; i++)
      {
      newnode = malloc(sizeof(struct node));
      p->next = newnode;
      printf("Enter number to be inserted : ");
      scanf("%d", &data);
      newnode->info = data;
      p = p->next;
      }
      p-> next= NULL;

}
void traverse()
{
   int i=1;
   struct node* p;
   if (head== NULL)
      printf("\nList is empty\n");
   else {
      p = head;
      while (p != NULL) {
         printf("Data %d= %d\n",i,p->info);
         p = p->next;
         i++;
      }
   }
}
void insertatfront()
{
   int data;
   struct node* p;
   p = malloc(sizeof(struct node));
   printf("Enter number to be inserted : ");
   scanf("%d", &data);
   p->next = head;
   head = p;
}
void insertatend()
{
   int data;
   struct node *p, *q;
   p = malloc(sizeof(struct node));
   printf("Enter number to be inserted : ");
   scanf("%d", &data);
   p->next = NULL;
   p->info = data;
   q= head;
   while (q->next != NULL) {
      q= q->next;
   }
   q->next = p;
}
void insertatpos()
{
   int pos,data;
   struct node *p,*q,*newnode;
   printf("Enter data point before which node to be inserted:");
   scanf("%d",&pos);
   printf("Enter no to be inserted:");
   scanf("%d",&data);
   q=head;
   while(q->info!=pos)
   {
      p=q;
      q=q->next;
   }
   newnode=malloc(sizeof(struct node));
   newnode->info= data;
   p->next=newnode;
   newnode->next=q;
}

int main()
{
   int c;
   printf("1.Create\n2.Display\n3.Insert at front\n4.Insert at end\n5.Insert at any
position\n6.Exit\n");
   do{
   printf("\nEnter choice:");
   scanf("%d",&c);
   switch(c)
   {
      case 1:createlist();
             break;
      case 2:traverse();
             break;
      case 3:insertatfront();
             break;
      case 4: insertatend();
             break;
      case 5:insertatpos();
             break;
      case 6 : exit(0);
      default: printf("Invalid choice");
   }
 }while(c>0);
 return 0;
}
```

OUTPUT



```
1.Create
2.Display
3.Insert at front
4.Insert at end
5.Insert at any position
6.Exit

Enter choice:1
Enter the number of nodes: 3
Enter number to be inserted : 2
Enter number to be inserted : 4
Enter number to be inserted : 6

Enter choice:3
Enter number to be inserted : 8

Enter choice:4
Enter number to be inserted : 10

Enter choice:5
Enter data point before which node to be inserted:4
Enter no to be inserted:12

Enter choice:2
Data 1= 0
Data 2= 2
Data 3= 12
Data 4= 4
Data 5= 6
Data 6= 10
```

PROGRAM 6

WAP to Implement Singly Linked List with following operations
a) Create a linked list.
b) Deletion of first element, specified element and last element in the list.
c) Display the contents of the linked list.

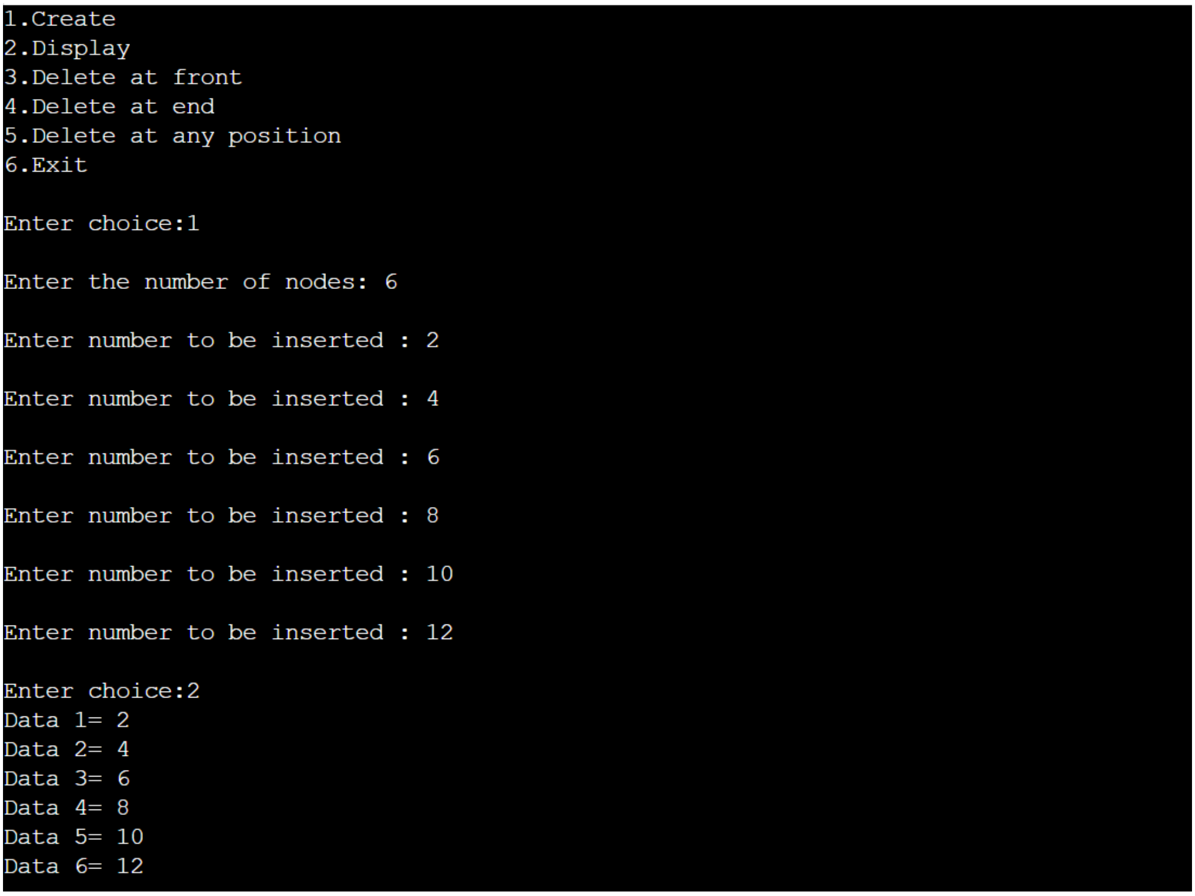
CODE

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

struct node* head = NULL;

void createlist()
{
   int n;
   printf("\nEnter the number of nodes: ");
   scanf("%d", &n);
   int data;
   struct node* newnode;
   struct node* p;
   newnode = malloc(sizeof(struct node));
   head = newnode;
   p = head;
   printf("\nEnter number to be inserted : ");
   scanf("%d", &data);
   head ->info = data;

   for (int i = 1; i < n; i++)
      {
      newnode = malloc(sizeof(struct node));
      p->next = newnode;
      printf("\nEnter number to be inserted : ");
      scanf("%d", &data);
      newnode->info = data;
      p = p->next;
      }
      p-> next= NULL;

}
void traverse()
{
   int i=1;
   struct node* p;
```

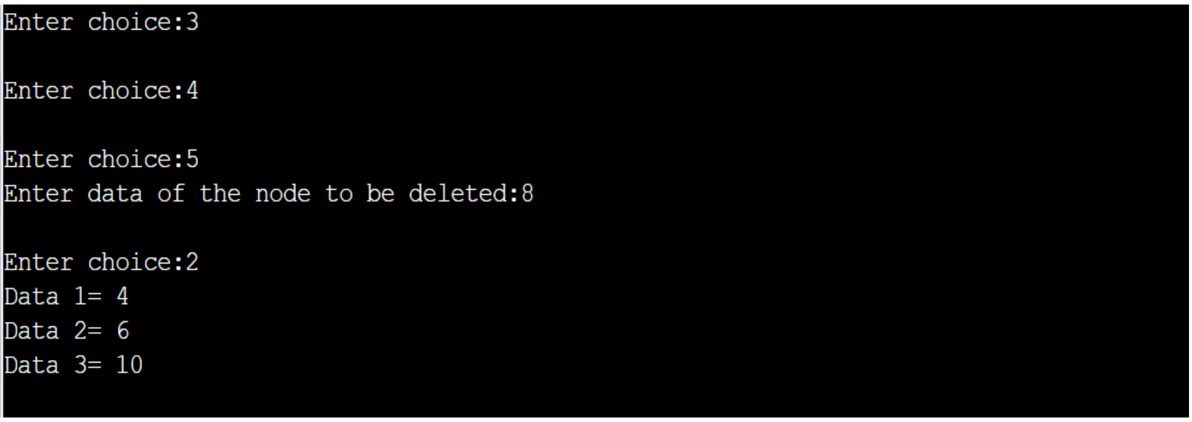
```
    if (head== NULL)
        printf("\nList is empty\n");
    else {
        p = head;
        while (p != NULL) {
            printf("Data %d= %d\n",i,p->info);
            p = p->next;
            i++;
        }
    }
}

void deleteatfront()
{
    struct node *p, *q;
    q=head;
    p=head->next;
    head=p;
    free(q);
}
void deleteatend()
{
    struct node *p,*q;
    q=head;
    while(q->next!= NULL)
    {
        p=q;
        q=q->next;
    }
    p->next=NULL;
    free(q);
}
void deleteatpos()
{
    struct node *p,*q;
    int pos;
    printf("Enter data of the node to be deleted:");
    scanf("%d",&pos);
    q=head;
    while(q->info!=pos)
    {
        p=q;
        q=q->next;
    }
     p->next=q->next;
     free(q);
}
int main()
{
    int c;
```

```
   printf("1.Create\n2.Display\n3.Delete at front\n4.Delete at end\n5.Delete at any
position\n6.Exit\n");
   do{
   printf("\nEnter choice:");
   scanf("%d",&c);
   switch(c)
   {
      case 1:createlist();
            break;
      case 2:traverse();
            break;
      case 3: deleteatfront();
            break;
      case 4: deleteatend();
            break;
      case 5: deleteatpos();
            break;
      case 6: exit(0);
      default: printf("Invalid choice");
   }
  }while(c>0);
  return 0;
}
```

OUTPUT



```
1.Create
2.Display
3.Delete at front
4.Delete at end
5.Delete at any position
6.Exit

Enter choice:1

Enter the number of nodes: 6

Enter number to be inserted : 2

Enter number to be inserted : 4

Enter number to be inserted : 6

Enter number to be inserted : 8

Enter number to be inserted : 10

Enter number to be inserted : 12

Enter choice:2
Data 1= 2
Data 2= 4
Data 3= 6
Data 4= 8
Data 5= 10
Data 6= 12
```



```
Enter choice:3

Enter choice:4

Enter choice:5
Enter data of the node to be deleted:8

Enter choice:2
Data 1= 4
Data 2= 6
Data 3= 10
```

PROGRAM 7

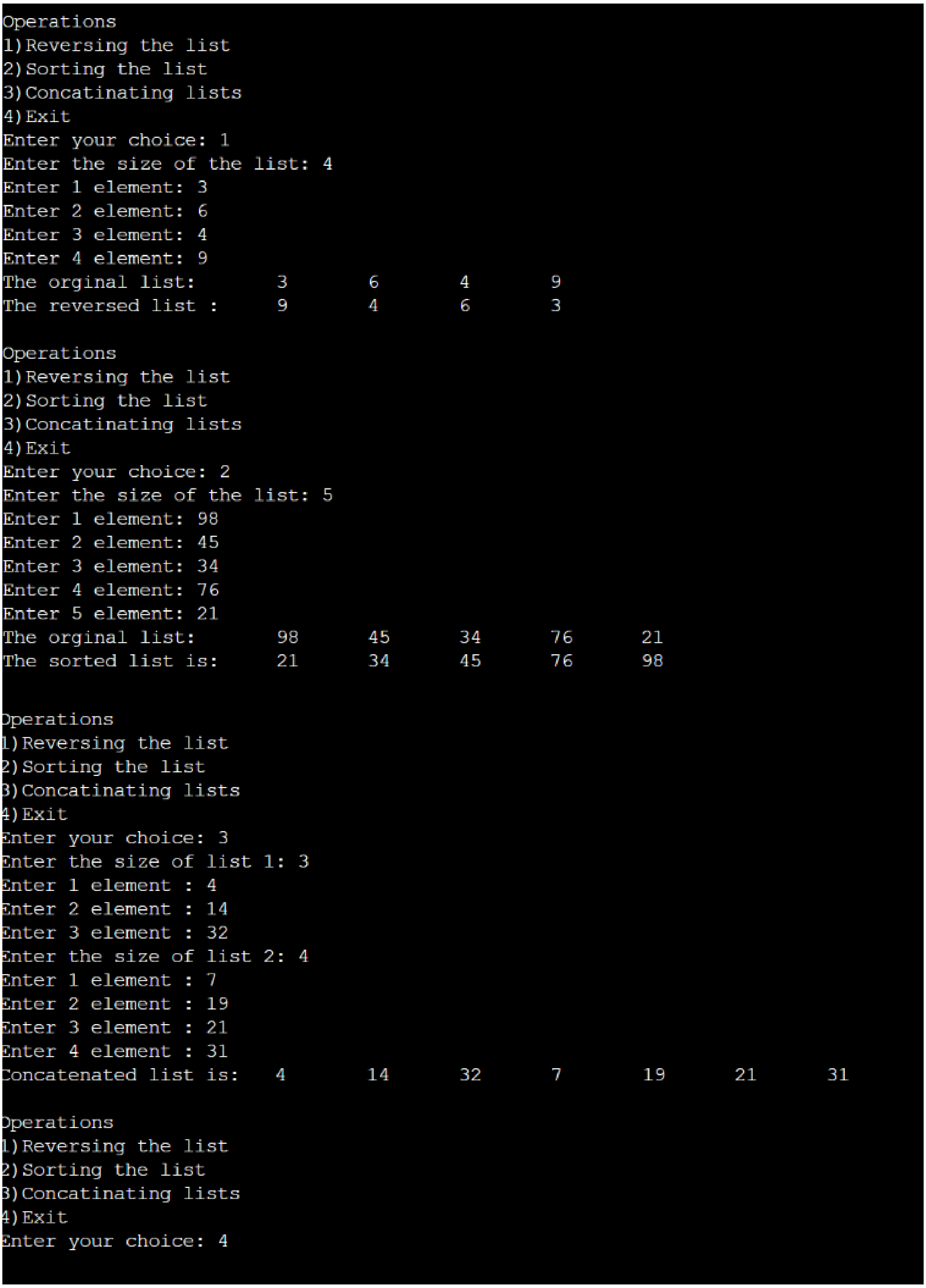
WAP to Implement Single Link List with following operations
a) Sort the linked list.
b) Reverse the linked list.
c) Concatenation of two linked lists

CODE

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

struct node{
    int data;
    struct node* next;
    struct node* prev;
};
struct node* head=NULL;

struct node3{
    int data;
    struct node3* next;
    struct node3* prev;
};
struct node3* head3=NULL;

struct node1{
    int data;
    struct node1 *next;
};
struct node1 *head1=NULL;
struct node1 *head2=NULL;

void rever_list();
void sort();
void concatination();
void main(){
    int ch;
    do{
        printf("\n\nOperations \n");
        printf("1)Reversing the list\n2)Sorting the list\n3)Concatinating lists\n4)Exit\n");
        printf("Enter your choice: ");
        scanf("%d",&ch);
        switch(ch){
            case 1:rever_list();
                   break;
            case 2:sort();
                  break;
            case 3:concatination();
                   break;
            case 4:exit(9);
        }
    }while(ch!=5);
}

void rever_list(){
    int n;
    printf("Enter the size of the list: ");
    scanf("%d",&n);
    for(int i=0; i<n; i++){
     struct node *temp;
     temp = (struct node*)malloc(sizeof(struct node));
     printf("Enter %d element: ",i+1);
     scanf("%d",&temp->data);
     temp->next = NULL;
     temp->prev = NULL;

     if(head==NULL){
         head = temp;
     }
     else{
         struct node* p;
         p = head;
         while(p->next != NULL){
             p = p->next;
         }
         p->next = temp;
         temp->prev = p;

     }

}
struct node* s, *t;
s = head;
t = head;
printf("The orginal list: ");
```

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

while(s->next != NULL){
     s = s->next;

}

printf("\nThe reversed list : ");
while(s != NULL){
    printf("\t%d",s->data);
    s = s->prev;
}


}
void sort(){
    int n;
    printf("Enter the size of the list: ");
    scanf("%d",&n);
    for(int i=0; i<n; i++){
     struct node3 *temp;
     temp = (struct node3*)malloc(sizeof(struct node3));
     printf("Enter %d element: ",i+1);
     scanf("%d",&temp->data);
     temp->next = NULL;
     temp->prev = NULL;

     if(head3==NULL){
         head3 = temp;
     }
     else{
         struct node3* p;
         p = head3;
         while(p->next != NULL){
             p = p->next;
         }
         p->next = temp;
         temp->prev = p;

     }

}
struct node3 *t;
t = head3;
printf("The orginal list: ");
while(t != NULL){
    printf("\t%d",t->data);
    t = t->next;
}


struct node3 *current=NULL, *index=NULL, *p;
int temp;

for(current=head3; current->next!=NULL; current=current->next){
    for(index=current->next; index!=NULL; index=index->next){
        if(current->data > index->data){
            temp = current->data;
            current->data = index->data;
            index->data = temp;
        }

    }
}
p = head3;
printf("\nThe sorted list is:");
while(p != NULL){
    printf("\t%d",p->data);
    p = p->next;

}


}

void concatination(){
    struct node1 *temp;
    int n,m;
    printf("Enter the size of list 1: ");
    scanf("%d",&n);
    for(int i=0; i<n; i++){
        temp = (struct node1 *)malloc(sizeof(struct node1));
        printf("Enter %d element : ",i+1);
        scanf("%d",&temp->data);
        temp->next = NULL;

     if(head1==NULL){
         head1 = temp;
     }
     else{
         struct node1* p;
         p = head1;
         while(p->next != NULL){
             p = p->next;
         }
         p->next = temp;
     }

    }
    printf("Enter the size of list 2: ");
    scanf("%d",&m);
    struct node1 *p;

    for(int i=0; i<m; i++) {
        p = (struct node1*)malloc(sizeof(struct node1));
        printf("Enter %d element : ",i+1);
        scanf("%d",&p->data);
        p->next = NULL;

     if(head2==NULL){
         head2 = p;
     }
     else{
         struct node1* p1;
         p1 = head2;
         while(p1->next != NULL){
             p1 = p1->next;
         }
         p1->next = p;


     }
    }
    if(head1->next == NULL){
        head1->next = head2;
    }
    else{
       struct node1 *x;
       x = head1;
       while(x->next != NULL){
           x = x->next;
       }
       x->next = head2;
    }

    struct node1 *r;
    r = head1;
    printf("Concatenated list is: ");
    while(r != NULL){
        printf("\t%d",r->data);
        r = r->next;

    }

}
```

OUTPUT



```
Operations
1)Reversing the list
2)Sorting the list
3)Concatinating lists
4)Exit
Enter your choice: 1
Enter the size of the list: 4
Enter 1 element: 3
Enter 2 element: 6
Enter 3 element: 4
Enter 4 element: 9
The orginal list:       3       6       4       9
The reversed list :     9       4       6       3

Operations
1)Reversing the list
2)Sorting the list
3)Concatinating lists
4)Exit
Enter your choice: 2
Enter the size of the list: 5
Enter 1 element: 98
Enter 2 element: 45
Enter 3 element: 34
Enter 4 element: 76
Enter 5 element: 21
The orginal list:       98      45      34      76      21
The sorted list is:     21      34      45      76      98

Operations
1)Reversing the list
2)Sorting the list
3)Concatinating lists
4)Exit
Enter your choice: 3
Enter the size of list 1: 3
Enter 1 element : 4
Enter 2 element : 14
Enter 3 element : 32
Enter the size of list 2: 4
Enter 1 element : 7
Enter 2 element : 19
Enter 3 element : 21
Enter 4 element : 31
Concatenated list is:   4       14      32      7       19      21      31

Operations
1)Reversing the list
2)Sorting the list
3)Concatinating lists
4)Exit
Enter your choice: 4
```

PROGRAM 8

WAP to implement Stack & Queues using Linked Representation

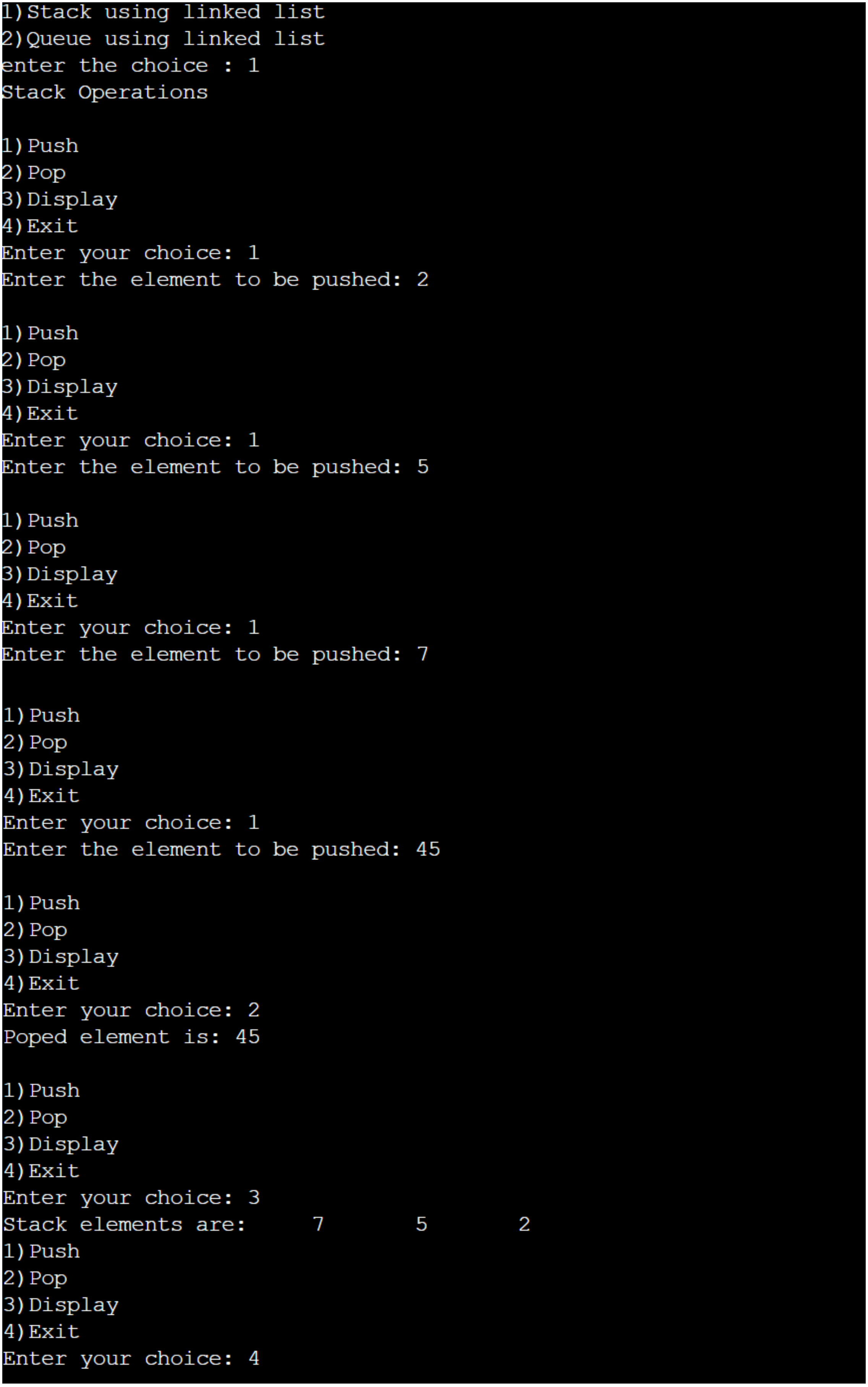
CODE

```
#include<stdio.h>
#include<stdlib.h>
void stack();
void queue();
void push();
int pop();
void display();
void enqueue();
int dequeue();
void display1();
struct node{
   int data;
   struct node* next;
};
struct node* top=NULL;

struct node1{
   int data;
   struct node1* next;
};
struct node1* front=NULL;
struct node1* rear=NULL;
void main(){
   int ch;
   printf("1)Stack using linked list\n2)Queue using linked list\n");
   printf("enter the choice : ");
   scanf("%d",&ch);
   switch(ch){
      case 1: stack();
           break;
      case 2: queue();
           break;
   }
}

void stack(){
   int ch,temp;
    printf("Stack Operations\n");
   while(ch != 5){
      printf("\n1)Push\n2)Pop\n3)Display\n4)Exit\n");
      printf("Enter your choice: ");
      scanf("%d",&ch);
      switch(ch){
         case 1: push();
              break;
```

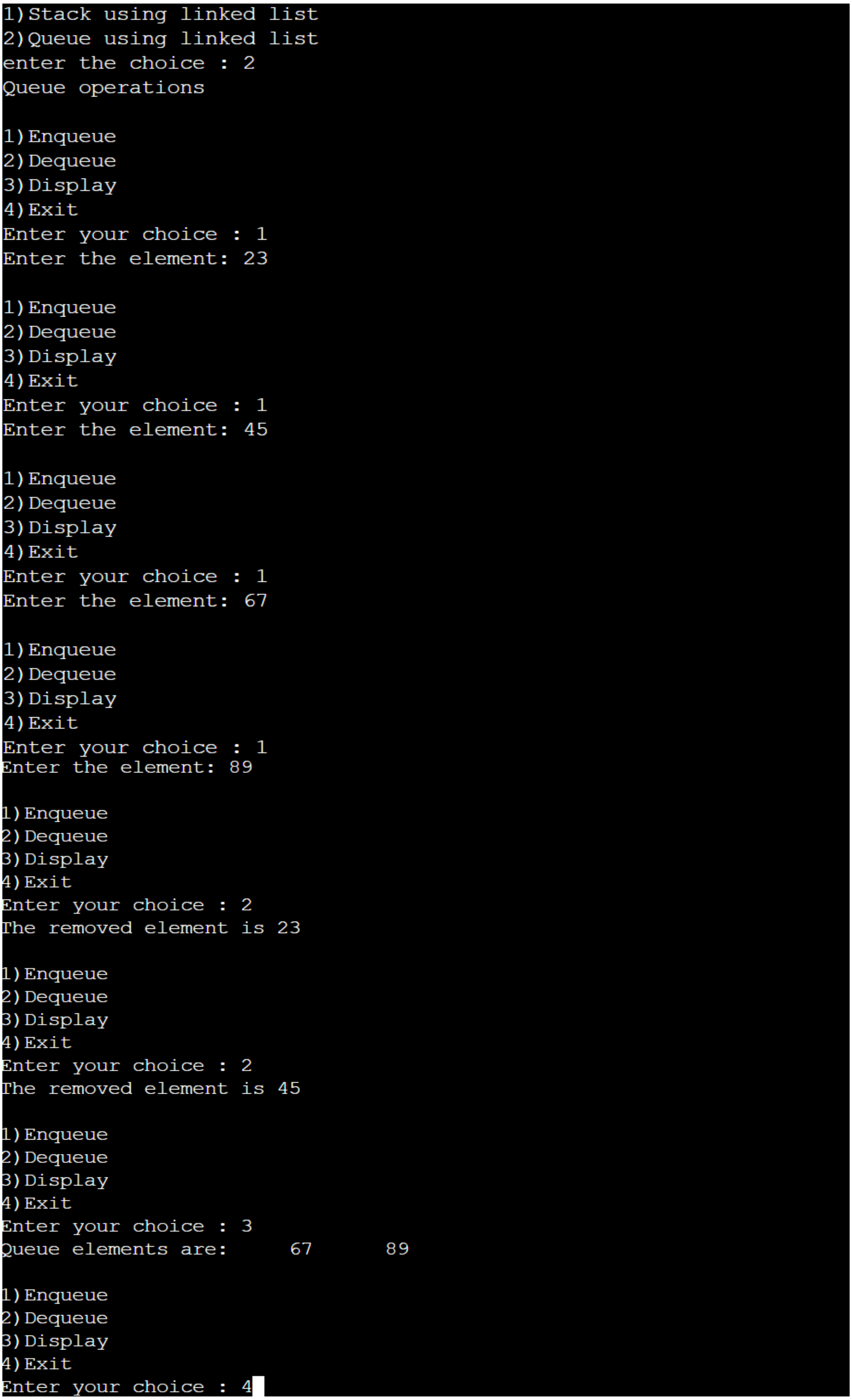
```
            case 2: temp = pop();
                    printf("Poped element is: %d\n",temp);
                    break;
            case 3: display();
                    break;
            case 4: exit(9);

        }
    }
}

void push(){
    struct node* temp;
    int n;
    temp = (struct node*)malloc(sizeof(struct node));
    printf("Enter the element to be pushed: ");
    scanf("%d",&n);
    temp->data = n;
    temp->next = top;

    top = temp;
}
int pop(){
    struct node *temp;
    temp = top;
    top = top->next;
    return temp->data;
    free(temp);
}
void display(){
    struct node *p;
    p = top;
    printf("Stack elements are:");
    while(p != NULL){
        printf("\t%d",p->data);
        p = p->next;
    }
}

void queue(){
    int ch,temp;
     printf("Queue operations\n");
    while(ch != 5){
        printf("\n1)Enqueue\n2)Dequeue\n3)Display\n4)Exit\n");
        printf("Enter your choice : ");
        scanf("%d",&ch);
        switch(ch){
            case 1: enqueue();
                    break;
            case 2: temp = dequeue();
```

```
                 printf("The removed element is %d\n",temp);
                break;
          case 3: display1();
                break;
          case 4: exit(9);

        }
    }
}
void enqueue(){
    struct node1 *temp;
    int n;
    temp = (struct node1 *)malloc(sizeof(struct node1));
    printf("Enter the element: ");
    scanf("%d",&n);
    temp->data = n;
    temp->next=NULL;

    if(front==NULL || rear==NULL){
        front=temp;
        rear=temp;
    }
    else{
        struct node1* p;
        p = front;
        while(p->next != NULL){
            p = p->next;
        }
        p->next = temp;
        rear = rear->next;
    }
    }

int dequeue(){
    struct node1* r;
    r = front;
    front = front->next;
     return r->data;
    free(r);
  }
void display1(){
    struct node1* p;
    p = front;
    printf("Queue elements are:");
    while(p != NULL){
        printf("\t%d",p->data);
        p = p->next;
    }
    printf("\n");
}
```

OUTPUT



```
1)Stack using linked list
2)Queue using linked list
enter the choice : 1
Stack Operations

1)Push
2)Pop
3)Display
4)Exit
Enter your choice: 1
Enter the element to be pushed: 2

1)Push
2)Pop
3)Display
4)Exit
Enter your choice: 1
Enter the element to be pushed: 5

1)Push
2)Pop
3)Display
4)Exit
Enter your choice: 1
Enter the element to be pushed: 7

1)Push
2)Pop
3)Display
4)Exit
Enter your choice: 1
Enter the element to be pushed: 45

1)Push
2)Pop
3)Display
4)Exit
Enter your choice: 2
Poped element is: 45

1)Push
2)Pop
3)Display
4)Exit
Enter your choice: 3
Stack elements are:     7       5       2
1)Push
2)Pop
3)Display
4)Exit
Enter your choice: 4
```



```
1)Stack using linked list
2)Queue using linked list
enter the choice : 2
Queue operations

1)Enqueue
2)Dequeue
3)Display
4)Exit
Enter your choice : 1
Enter the element: 23

1)Enqueue
2)Dequeue
3)Display
4)Exit
Enter your choice : 1
Enter the element: 45

1)Enqueue
2)Dequeue
3)Display
4)Exit
Enter your choice : 1
Enter the element: 67

1)Enqueue
2)Dequeue
3)Display
4)Exit
Enter your choice : 1
Enter the element: 89

1)Enqueue
2)Dequeue
3)Display
4)Exit
Enter your choice : 2
The removed element is 23

1)Enqueue
2)Dequeue
3)Display
4)Exit
Enter your choice : 2
The removed element is 45

1)Enqueue
2)Dequeue
3)Display
4)Exit
Enter your choice : 3
Queue elements are:     67      89

1)Enqueue
2)Dequeue
3)Display
4)Exit
Enter your choice : 4
```

PROGRAM 9

WAP to 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

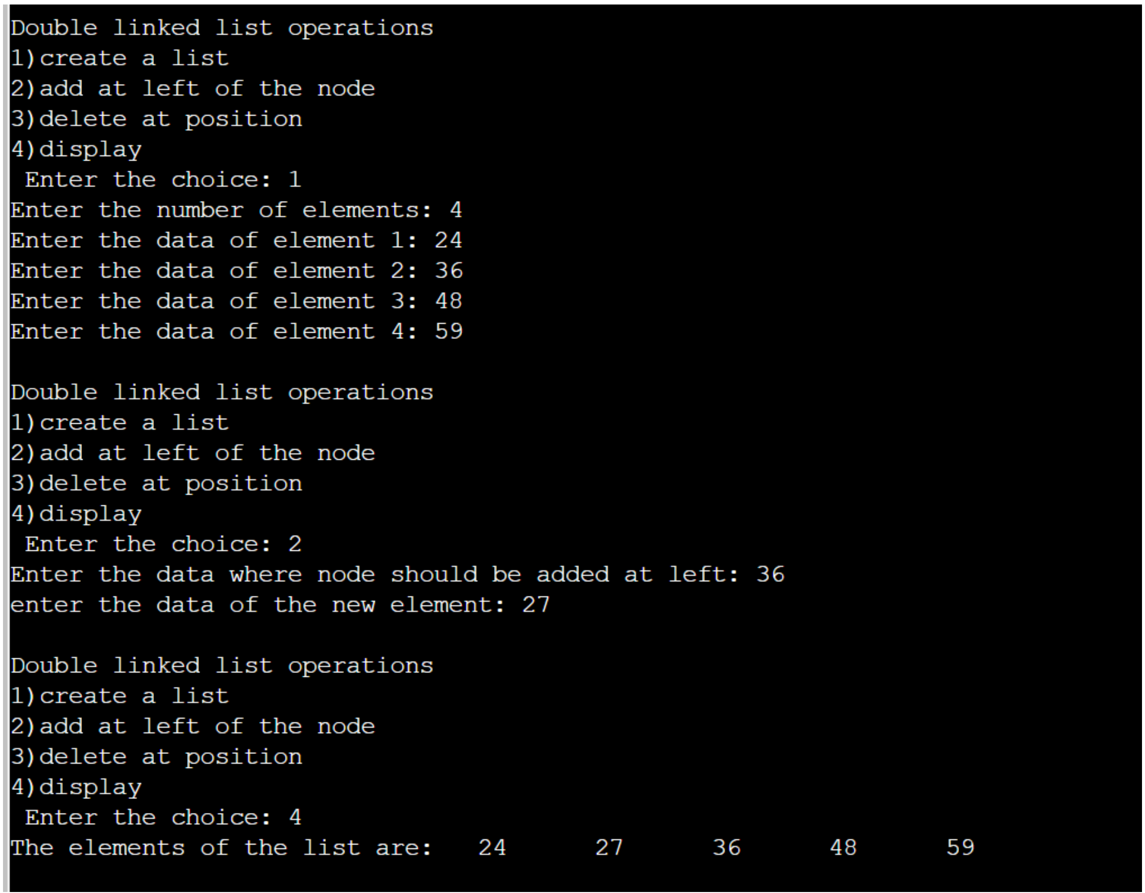
CODE

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

struct node* head = NULL;

void append();
void add_at_left();
void delete_at_pos();
void display();

void main()
{
   int ch;
   while(ch != 5){
      printf("\nDouble linked list operations\n");
      printf("1)create a list\n2)add at left of the node\n3)delete at position\n4)display\n ");
      printf("Enter the choice: ");
      scanf("%d",&ch);

      switch(ch){
      case 1: append();
            break;
      case 2: add_at_left();
            break;
      case 3: delete_at_pos();
            break;
      case 4: display();
            break;
      case 5:exit(0);

      }
   }
}
```

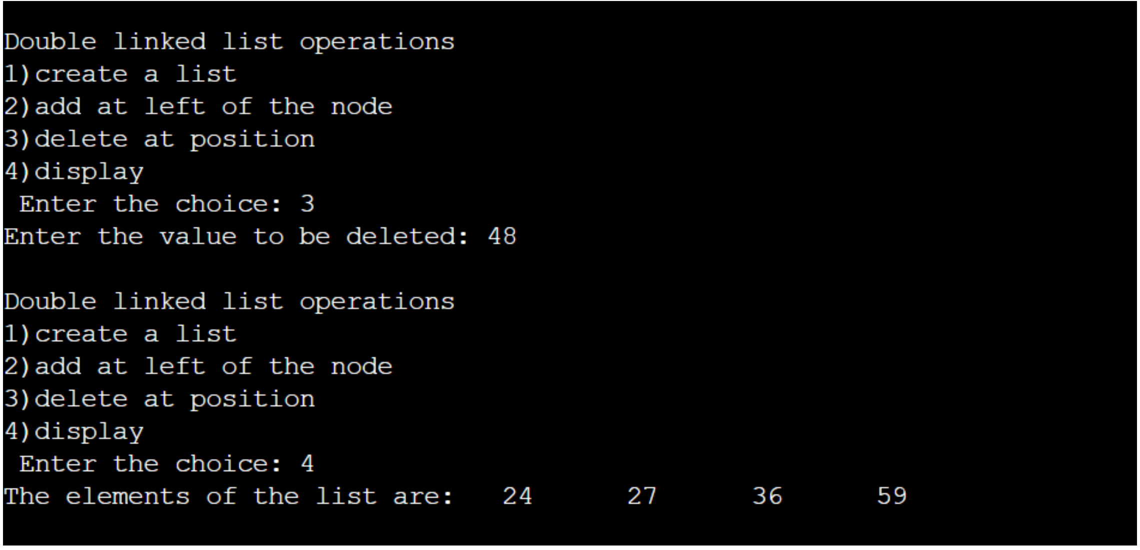
```
void append()
{
    int n;
    printf("Enter the number of elements: ");
    scanf("%d",&n);
    for(int i=1; i<=n; i++){
    struct node* temp;
    temp = (struct node*)malloc(sizeof(struct node));
    printf("Enter the data of element %d: ",i);
    scanf("%d",&temp->data);
    temp->next = NULL;
    temp->prev = NULL;

    if(head == NULL){
        head = temp;
    }
    else{
        struct node* p;
        p = head;
        while(p->next != NULL){
            p = p->next;
        }
        p->next = temp;
        temp->prev = p;
    }

    }
}

void display()
{
    if(head == NULL){
        printf("The list is empty\n");
    }

    else{
    struct node* p;
    p = head;

   printf("The elements of the list are: ");
    while(p != NULL){
        printf("\t%d",p->data);
        p = p->next;
    }
    printf("\n");
    }

}
```

```
void add_at_left()
{
   int val;
   printf("Enter the data where node should be added at left: ");
   scanf("%d",&val);

   struct node* temp,*p,*r;
   temp = (struct node*)malloc(sizeof(struct node));
   printf("enter the data of the new element: ");
   scanf("%d",&temp->data);
   temp->next = NULL;
   temp->prev = NULL;
   p = head;
   while(p->data != val){
      p = p->next;
   }
   r = p->prev;

   temp->next = p;
   p->prev = temp;
   r->next = temp;
   temp->prev = r;
}

void delete_at_pos()
{
   int val;
   printf("Enter the value to be deleted: ");
   scanf("%d",&val);

   struct node *temp, *p;
   p = head;
   while(p->data != val){
      p = p->next;
   }

   p->prev->next = p->next;
   p->next->prev = p->prev;
   p->next = NULL;
   p->prev = NULL;
}
```

OUTPUT



```
Double linked list operations
1)create a list
2)add at left of the node
3)delete at position
4)display
 Enter the choice: 1
Enter the number of elements: 4
Enter the data of element 1: 24
Enter the data of element 2: 36
Enter the data of element 3: 48
Enter the data of element 4: 59

Double linked list operations
1)create a list
2)add at left of the node
3)delete at position
4)display
 Enter the choice: 2
Enter the data where node should be added at left: 36
enter the data of the new element: 27

Double linked list operations
1)create a list
2)add at left of the node
3)delete at position
4)display
 Enter the choice: 4
The elements of the list are:   24      27      36      48      59
```



```
Double linked list operations
1)create a list
2)add at left of the node
3)delete at position
4)display
 Enter the choice: 3
Enter the value to be deleted: 48

Double linked list operations
1)create a list
2)add at left of the node
3)delete at position
4)display
 Enter the choice: 4
The elements of the list are:   24      27      36      59
```

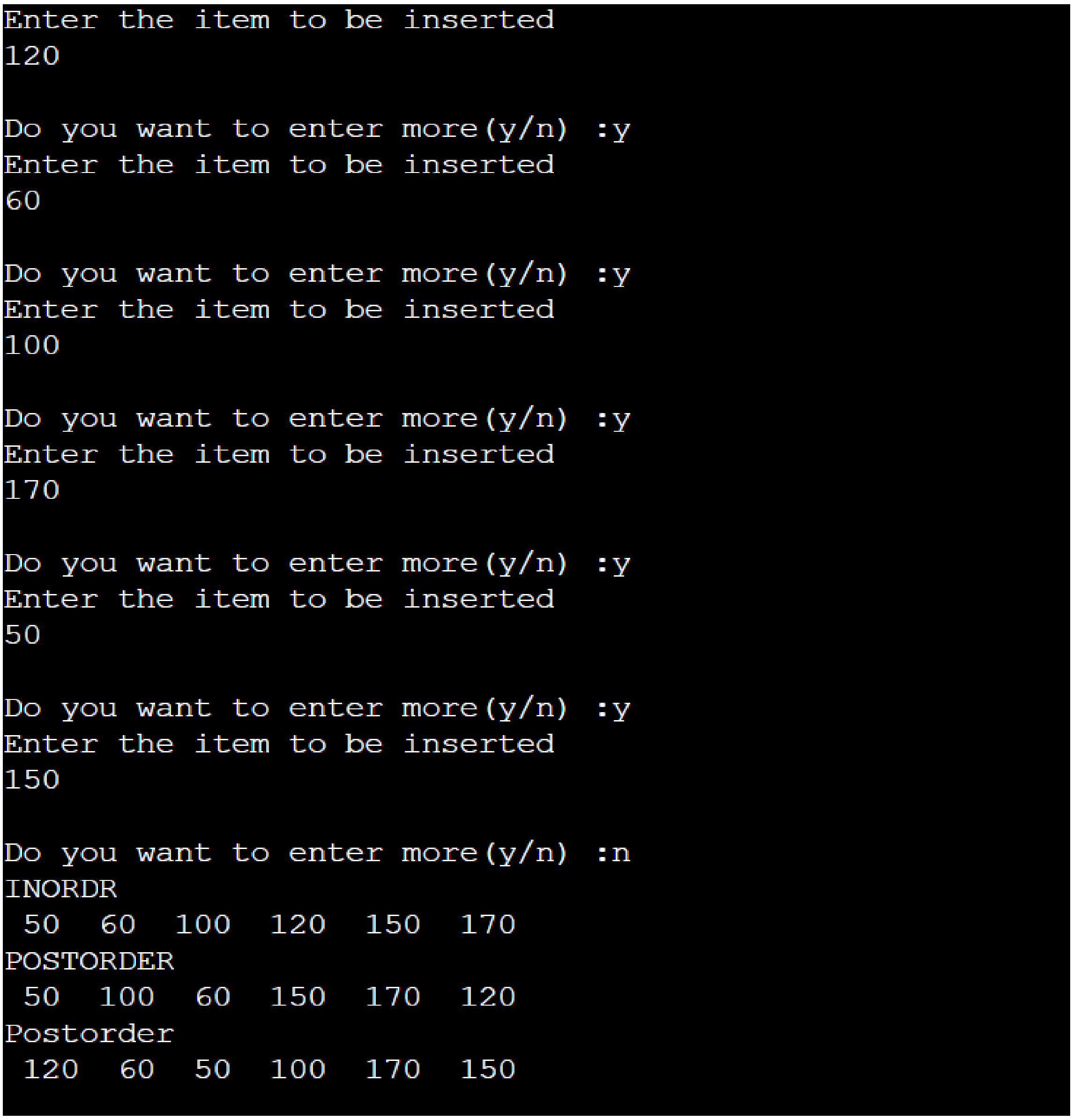
PROGRAM 10

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.

CODE

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
 struct node *left;
 struct node *right;
 int value;
};
typedef struct node* NODE;
NODE getnode()
{
 NODE temp;
 temp=(NODE)malloc(sizeof(struct node));
 if(temp==NULL)
 {
 printf("Memory not allocated\n");
 return NULL;
 }
 printf("Enter the item to be inserted\n");
 scanf("%d",&temp->value);
 temp->left=NULL;
 temp->right=NULL;
 return temp;
}
void insert(NODE root,NODE temp)

{
 if(temp->value<root->value)
 {
 if(root->left!=NULL)
 insert(root->left,temp);
 else
 root->left=temp;
 }
 if(temp->value>root->value)
 {
 if(root->right!=NULL)
 insert(root->right,temp);
 else
 root->right=temp;
 }
}
void traverseInorder(NODE root)
{
 if (root == NULL)
 return;
 traverseInorder(root->left);
 printf(" %d ", root->value);
 traverseInorder(root->right);
}
void traversePreorder(NODE root)
{
 if (root == NULL)
 return;
 printf(" %d ", root->value);
 traversePreorder(root->left);
 traversePreorder(root->right);
}

void traversePostorder(NODE root)
{
 if (root == NULL)
 return;
 traversePostorder(root->left);
 traversePostorder(root->right);
 printf(" %d ", root->value);
}
void main()
{
 NODE root=NULL,temp=NULL;
 char ch;
 do
 {
 temp=getnode();
 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');

 printf("INORDR\n");
 traverseInorder(root);
 printf("\n");
 printf("POSTORDER\n");
 traversePostorder(root);
 printf("\n");
 printf("Postorder\n");
 traversePreorder(root);
 printf("\n");
}
```

<u>OUTPUT</u>



```
Enter the item to be inserted
120

Do you want to enter more(y/n) :y
Enter the item to be inserted
60

Do you want to enter more(y/n) :y
Enter the item to be inserted
100

Do you want to enter more(y/n) :y
Enter the item to be inserted
170

Do you want to enter more(y/n) :y
Enter the item to be inserted
50

Do you want to enter more(y/n) :y
Enter the item to be inserted
150

Do you want to enter more(y/n) :n
INORDR
 50  60  100  120  150  170
POSTORDER
 50  100  60  150  170  120
Postorder
 120  60  50  100  170  150
```