VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
on

DATA STRUCTURES

Submitted by

AKSHANTH H M(1BM21CS014)

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
Oct 2022-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**" carried out by **Akshanth.H.M(1BM21CS014),** who is a 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 2022-23. The Lab report has been approved as it satisfies the academic requirements in respect of Data structures Lab **- (22CS3PCDST)**work prescribed for the said degree.

Name of the Lab-Incharge
Designation
Department of CSE
BMSCE, Bengaluru

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

`

Index Sheet

	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	
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 postorder c) To display the elements in the tree.	

Course Outcome

LAB PROGRAM 1:

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

a) Push

b) Pop

c) Display

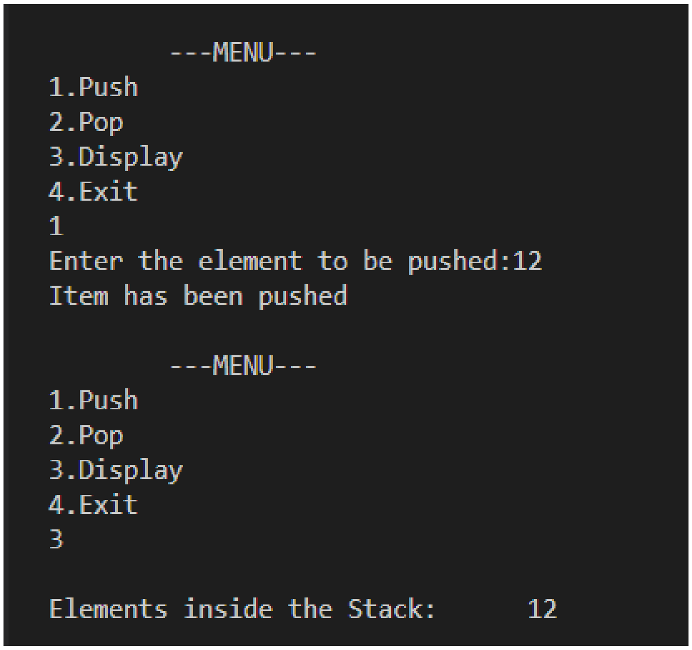
The program should print appropriate messages for stack overflow, stack underflow.

Code:

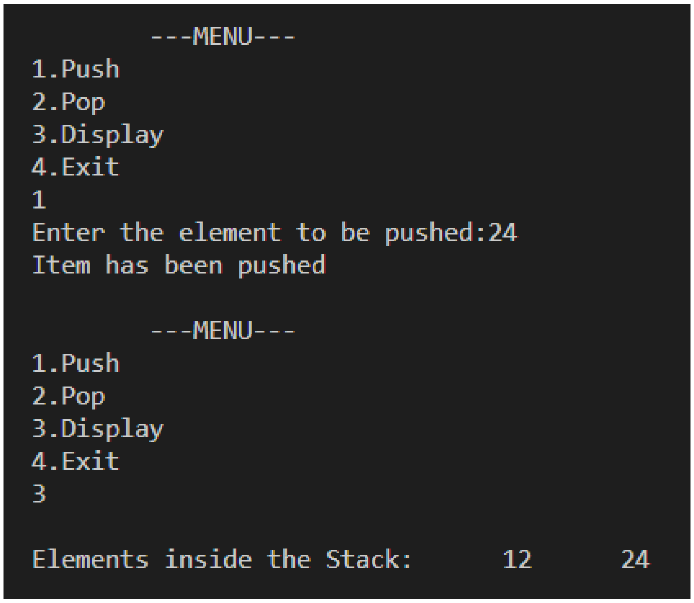
```
#include <stdio.h>

#include <stdlib.h>

#define SIZE 100


void push(int stack[],int *top,int *ptr)

{

   if(*top==SIZE)

   {

      printf("Overflow");

   }

   else

   {

   stack[++(*top)]=*ptr;

   }

}
```

```
int pop(int *top,int stack[])
{
    int del_item;
    if(*top==-1)
    {
    printf("\nUnderflow");
    }
    else
    {
        del_item=stack[*top];
        (*top)--;
        return del_item;
    }
}

void display(int *top,int stack[])
{
    int i;
    if(*top==-1)
    printf("\nUnderflow");
    else
    {
        printf("\nElements inside the Stack:");
        for(i=0;i<=(*top);i++)
        {
        printf("\t%d",stack[i]);
        }
    }
}

int main()
{
    int choice,l,item,stack[SIZE],top=-1;

while(1)
{
    printf("\n\n\t---MENU---\n");
    printf("1.Push\n2.Pop\n3.Display\n4.Exit\n");
    scanf("%d",&choice);

    switch(choice)
    {
        case 1: printf("Enter the element to be pushed:");
                scanf("%d",&item);
                push(stack,&top,&item);
                printf("Item has been pushed");
                break;
        case 2: l=pop(&top,stack);
                printf("Removed Item:%d",l);
                break;
        case 3: display(&top,stack);
                break;
        case 4: exit(0);
                break;
    }
}
    return 0;
}
```

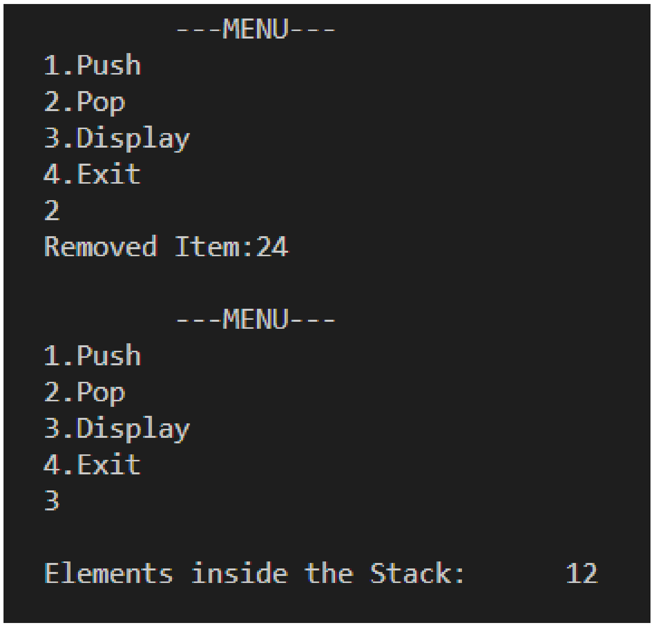
Output:



```
        ---MENU---
1.Push
2.Pop
3.Display
4.Exit
1
Enter the element to be pushed:12
Item has been pushed

        ---MENU---
1.Push
2.Pop
3.Display
4.Exit
3

Elements inside the Stack:      12
```



```
        ---MENU---
1.Push
2.Pop
3.Display
4.Exit
1
Enter the element to be pushed:24
Item has been pushed

        ---MENU---
1.Push
2.Pop
3.Display
4.Exit
3

Elements inside the Stack:      12      24
```



```
        ---MENU---
1.Push
2.Pop
3.Display
4.Exit
2
Removed Item:24

        ---MENU---
1.Push
2.Pop
3.Display
4.Exit
3

Elements inside the Stack:      12
```

LAB 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).

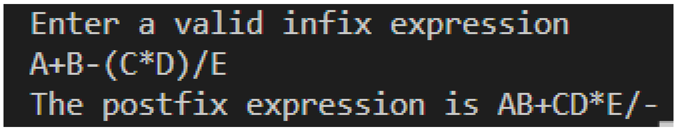
Code:

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

int top=-1;
char s[20];
char infix[20];
char postfix[20];
void inf_to_post();
int sp(char);
int ip(char);
void push(char);
char pop();

void main()
{
   printf("Enter a valid infix expression\n");
   scanf("%s",infix);
   inf_to_post();
   printf("The postfix expression is %s",postfix);
}
```

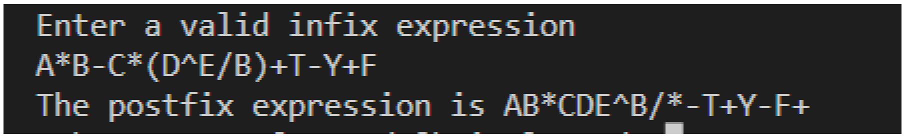
```
void push(char item)
{
   s[++top]=item;
}

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

int sp(char item)
{
   switch(item)
{
   case '+':
   case '-': return 2;
   case '*':
   case '/': return 4;
   case '^':
   case '$': return 5;
   case '(': return 0;
   case '#': return -1;
   default: return 8;
}
}
```

```
int ip( char item)

{

switch(item)

{

   case '+':

   case '-': return 1;

   case '*':

   case '/': return 3;

   case '^':

   case '$':return 6;

   case '(':return 9;

   case ')': return 0;

   default: return 7;

}

}


void inf_to_post()

{

   int i,j=0;

   char symbol;

   push('#');

for(i=0;i<strlen(infix);i++)

{

   symbol=infix[i];


while(sp(s[top])>ip(symbol))
{
    postfix[j]=pop();
    j++;
}

if(sp(s[top])<ip(symbol))
{
    push(symbol);
}

if(sp(s[top])==ip(symbol))
{
    pop();
}
}



while(s[top]!='#')
{
    postfix[j]=pop();
    j++;
}
    postfix[j]='\0';
}
```

Output:



```
Enter a valid infix expression
A+B-(C*D)/E
The postfix expression is AB+CD*E/-
```



```
Enter a valid infix expression
A*B-C*(D^E/B)+T-Y+F
The postfix expression is AB*CDE^B/*-T+Y-F+
```

LAB 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

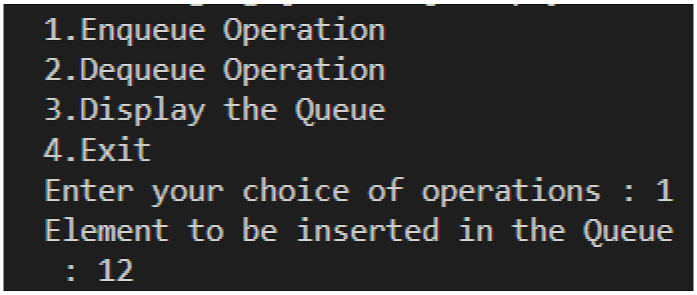
Code:

```
#include <stdio.h>
#include<stdlib.h>
# define SIZE 100
void enqueue();
void dequeue();
void show();
int arr[SIZE];
int Rear = - 1;
int Front = - 1;
int main()
{
   int ch;
   while (1)
```

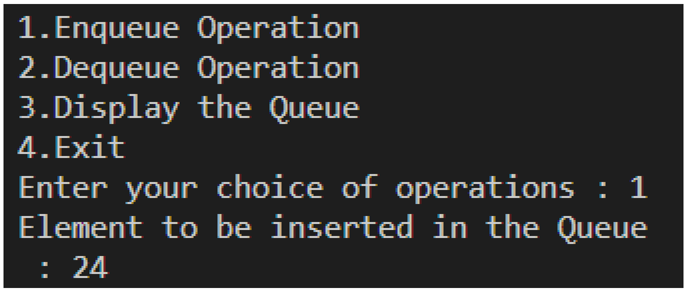
```
    {
        printf("1.Enqueue Operation\n");
        printf("2.Dequeue Operation\n");
        printf("3.Display the Queue\n");
        printf("4.Exit\n");
        printf("Enter your choice of operations : ");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1:
            enqueue();
            break;
            case 2:
            dequeue();
            break;
            case 3:
            show();
            break;
            case 4:
            exit(0);
            default:
            printf("Incorrect choice \n");
        }
    }
    return 0;
}

void enqueue()
```

```
{
    int insert_item;
    if (Rear == SIZE - 1)
       printf("Overflow \n");
    else
    {
        if (Front == - 1)

        Front = 0;
        printf("Element to be inserted in the Queue\n : ");
        scanf("%d", &insert_item);
        Rear = Rear + 1;
        arr[Rear] = insert_item;
    }
}

void dequeue()
{
    if ((Front == - 1) || (Front > Rear))
    {
        printf("Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from the Queue: %d\n", arr[Front]);
        Front = Front + 1;
    }
}

void show()
{

    if (Front == - 1)
        printf("Empty Queue \n");
    else
    {
        printf("Queue: \n");
        for (int i = Front; i <= Rear; i++)
            printf("%d ", arr[i]);
        printf("\n");
    }
}
```

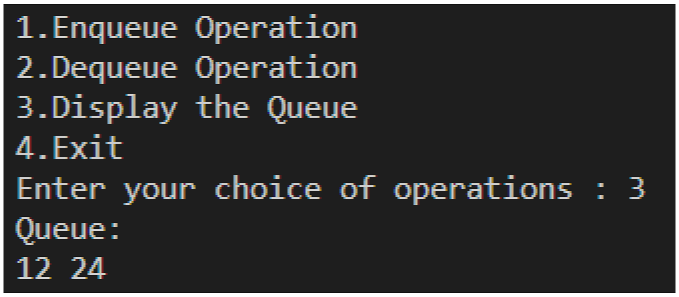
Output:



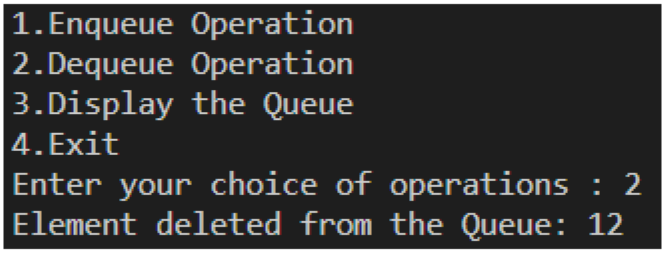
```
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 1
Element to be inserted in the Queue
 : 12
```



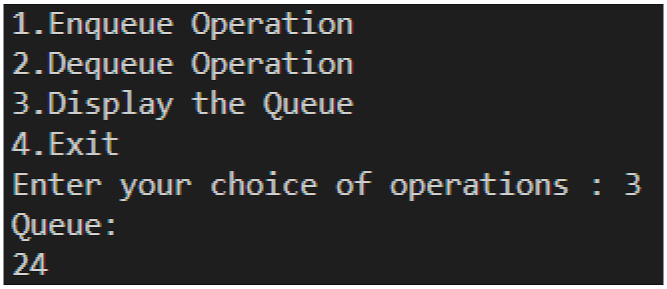
```
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 1
Element to be inserted in the Queue
 : 24
```



```
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 3
Queue:
12 24
```



```
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 2
Element deleted from the Queue: 12
```



```
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 3
Queue:
24
```

LAB 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.

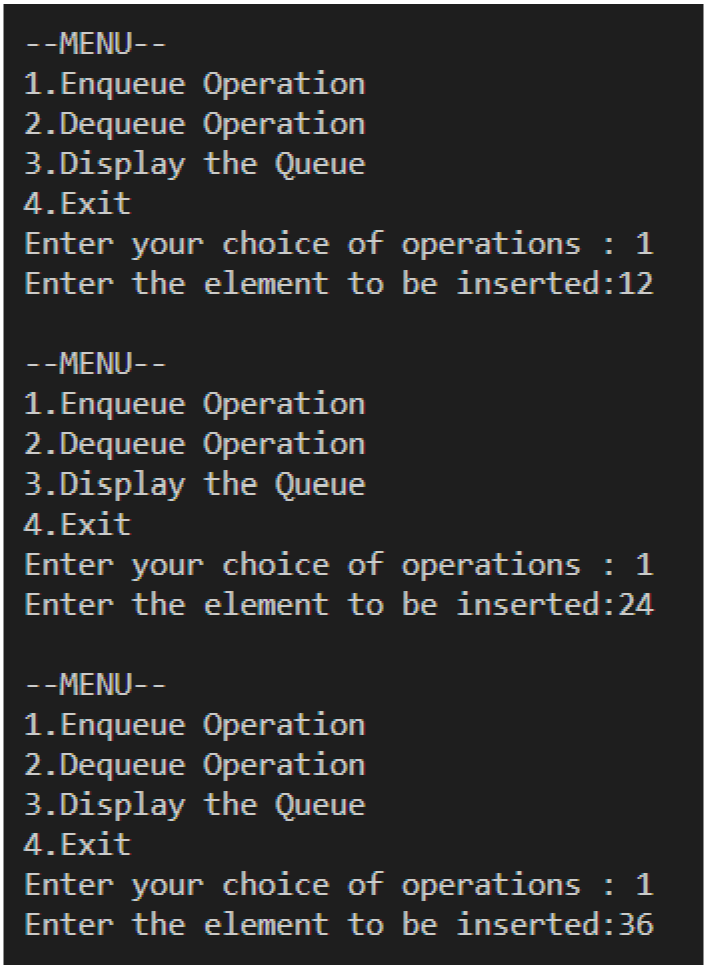
Code:

```
#include <stdio.h>

#include<stdlib.h>

# define SIZE 10


int count=0;

void enqueue(int,int [],int*,int*);

int dequeue(int [],int*,int*);

void display(int [],int*,int*);




int main()

{


int ch,ele,queue[SIZE];

int front=0;

int rear=-1;
```

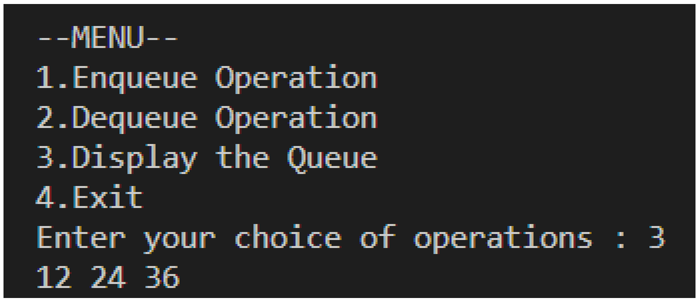
```
while (1)
{   printf("\n--MENU--\n");
    printf("1.Enqueue Operation\n");
    printf("2.Dequeue Operation\n");
    printf("3.Display the Queue\n");
    printf("4.Exit\n");
    printf("Enter your choice of operations : ");
    scanf("%d", &ch);
    switch (ch)
    {
        case 1:
        printf("Enter the element to be inserted:");
        scanf("%d",&ele);
        enqueue(ele,queue,&front,&rear);
        break;
        case 2:
        dequeue(queue,&front,&rear);
        printf("\nElement has been removed from the queue.");
        break;
        case 3:
        display(queue,&front,&rear);
        break;
        case 4:
        exit(0);
        default:
        printf("Incorrect choice \n");
    }
}
```

```
    return 0;



}


void enqueue(int E,int q[],int *f,int *r)
{
    if(count==SIZE)
    {
        printf("\nQueue is gonna overflowing");

    }
    else
    {   (*r)=((*r))%SIZE;
        (*r)++;
        q[(*r)]=(E);
        count++;
    }


}


int dequeue(int q[],int *f,int *r)
{
    int del_item;
    if(count==0)
    {
        printf("\nQueue does not have any element yet!");

    }
    else
    {
        del_item=q[(*f)];
        (*f)=((*f))%SIZE;
        (*f)++;
        count–;
        return(del_item);
    }




}

void display(int q[],int *f,int *r)
{
    int i,temp;
    temp=(*f);

    for(i=0;i<count;i++)
    {
        printf("%d ",q[temp]);
        temp=(temp)%SIZE;

        temp++;



    }




}
```

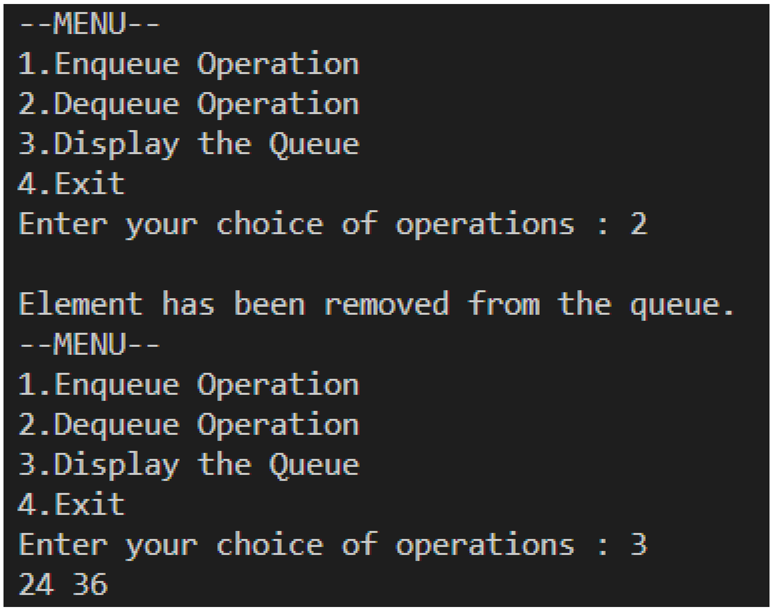
Output:



```
--MENU--
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 1
Enter the element to be inserted:12

--MENU--
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 1
Enter the element to be inserted:24

--MENU--
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 1
Enter the element to be inserted:36
```



```
--MENU--
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 3
12 24 36
```



```
--MENU--
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 2

Element has been removed from the queue.
--MENU--
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 3
24 36
```

LAB 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 value;

struct node *next;

};



typedef struct node *NODE;


//For allocating memory space-


NODE getnode()

{

   NODE temp;

   temp=(NODE)malloc(sizeof(struct node));


   if(temp==NULL)

   {
       printf("Memory could not be allocated.");
   }

   return(temp);

}



//Inserting values at the beginning-


NODE insert_beg(NODE first,int item)
{
   NODE temp;
   temp=getnode();

   temp->value=item;
   temp->next=NULL;

   if(first==NULL)
       return temp;
   else
   {
       temp->next=first;
       first=temp;
       return first;
   }

}
```

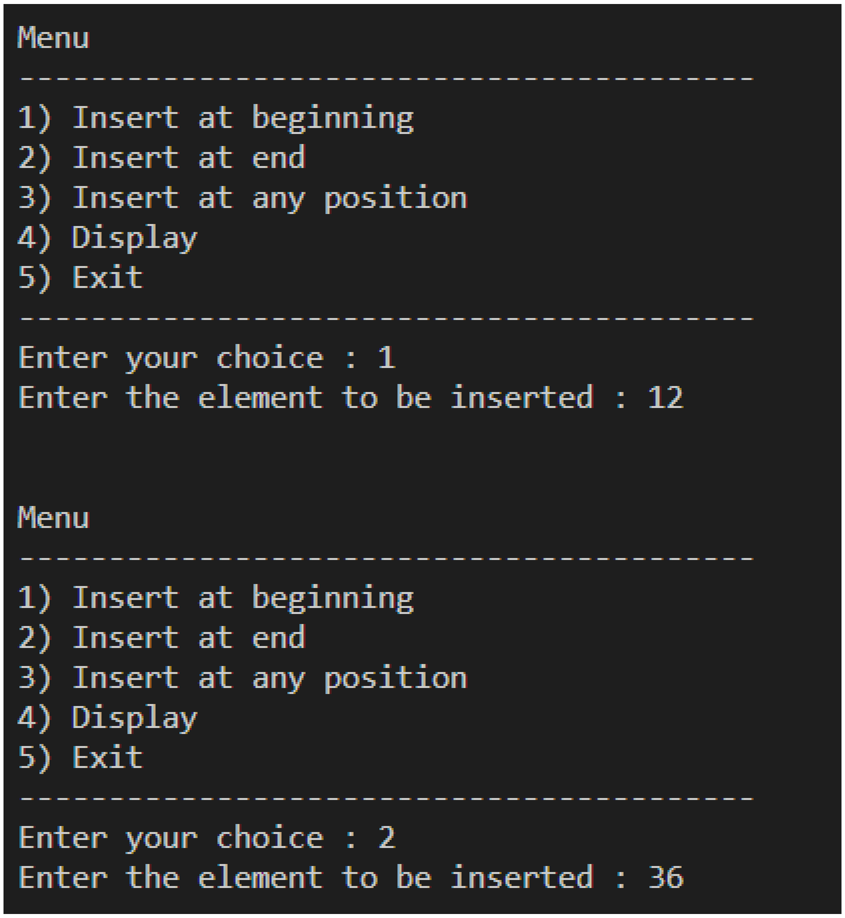
```
//Inserting at the end-

NODE insert_end(NODE first,int item)
{
    NODE New,last;
    New=getnode();
    New->value=item;
    New->next=NULL;

    if(first==NULL)
        return New;

    if(first->next==NULL)
    {
    first->next=New;
    return first;
    }

    last=getnode();
    last=first;

    while(last->next!=NULL)
        last=last->next;

    last->next=New;
    return first;

}



//For deleting at the beginning-

NODE delete_beg(NODE first)
{
    NODE temp;
    // temp=getnode();

    if(first==NULL)
    {
        printf("There is nothing to delete");
        return NULL;
    }

    temp=first;
    temp=temp->next;

    printf("Item has been deleted:%d\n",first->value);
    free(first);

    return(temp);
}
```

```
//For deleting at the end-



NODE delete_end(NODE first)
{
   NODE prev,curr;

   if(first==NULL)
   {
      printf("Nothing to delete");
      return NULL;
   }

   // prev=getnode();
   // curr=getnode();

   prev=NULL;
   curr=first;

   while(curr->next!=NULL)
   {
      prev=curr;
      curr=curr->next;
   }

   prev->next=NULL;
   free(curr);
   return(first);
```

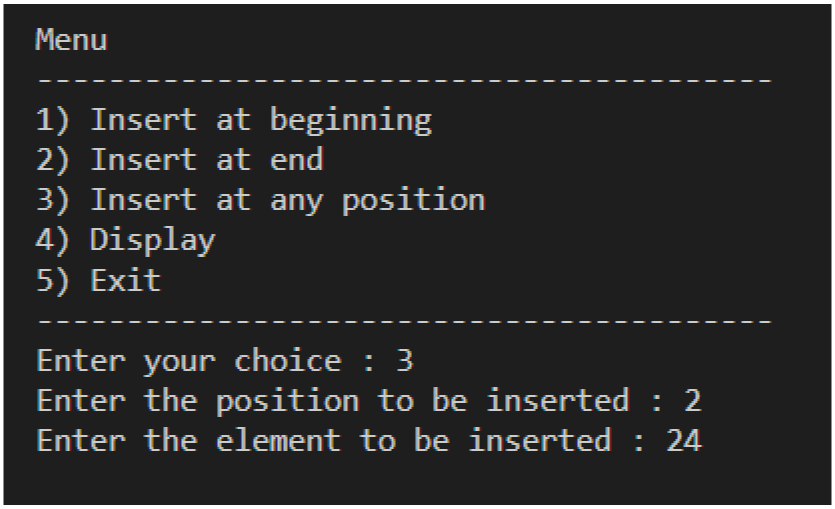
```
}



NODE insert_at_any_position(int item, int pos, NODE first)
{
    NODE curr,newnode,prev;
    int count=1;

    newnode=getnode();
    newnode->value=item;
    newnode->next=NULL;


    if((first==NULL)&&(pos==1)){
         printf("HI");
        return newnode;
    }

    if(first!=NULL && pos==1){
        newnode->next = first;
        first = newnode;
        return first;
    }

    prev=NULL;
    curr=first;
    while((count!=pos)&&(curr!=NULL))
    {
        prev=curr;
        curr=curr->next;
        count++;

    }

    if(count==pos)
    {
        prev->next=newnode;
        newnode->next=curr;
        return first;
    }

    if(curr==NULL)
    {
        printf("Position not found");
        return first;
    }

    return first;

}
```

```
//For displaying-

void display(NODE first)
{
    NODE temp;
    // temp=getnode();
    temp=first;

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


int main()
{
    int choice, item, x;
    NODE first = NULL;
    while (1)
    {
        printf("\n\nMenu\n-------------------------------------\n1) Insert at beginning\n2) Insert at
end\n3) Display\n4) Delete at Beginning\n5) Delete at the end\n6) Exit\n-----------------------
---------------\nEnter your choice : ");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

           printf("Enter the element to be inserted : ");

           scanf("%d", &x);

           first = insert_beg(first,x);

           break;

        case 2:

           printf("Enter the element to be inserted : ");

           scanf("%d", &x);

           first = insert_end(first,x);

           break;

        case 3:printf("/nThe items are:");

           display(first);

           break;

        case 4:

           first = delete_beg(first);

           printf("Item has been removed from the first NODE");

           break;

        case 5:

           first = delete_end(first);

           printf("Item has been removed from the last node.");

           break;

        case 6:
           printf("Enter the element to be inserted : ");
           scanf("%d", &pos);
           printf("Enter the element to be inserted : ");
           scanf("%d", &x);
           insert_at_any_position(pos, x, first);
           break;

        default:
           exit(0);
           break;
        }
    }
    return 0;
}
```

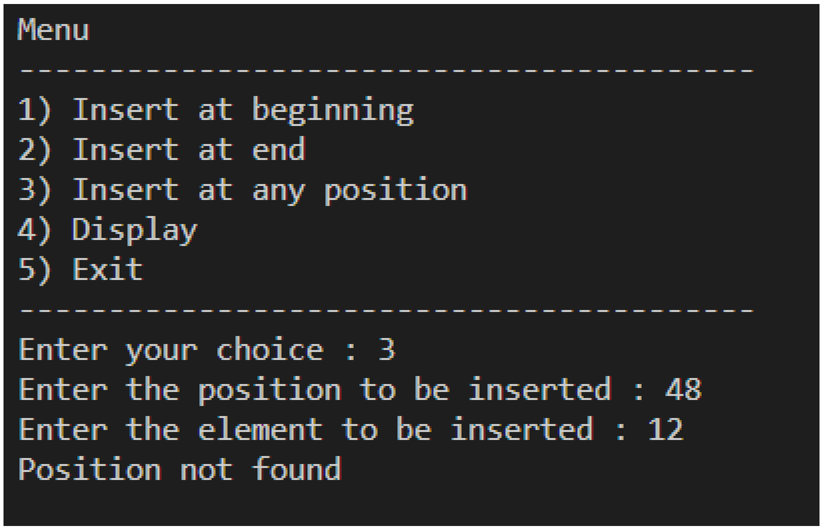
Output:



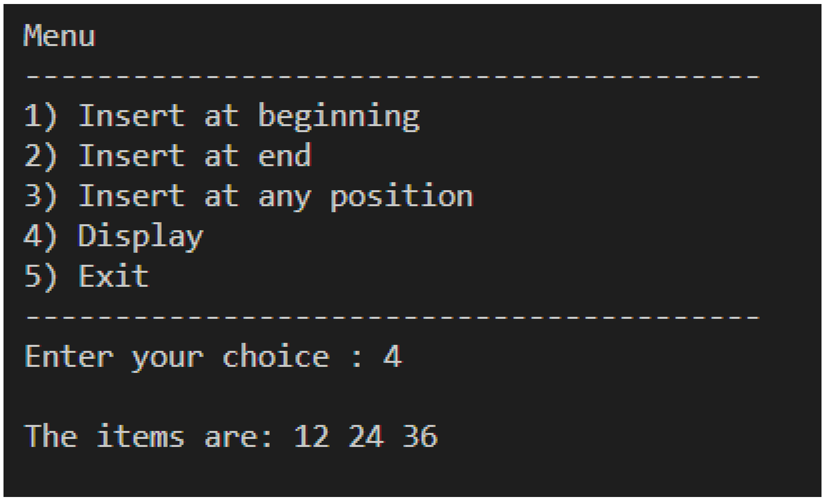
```
Menu
-----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Display
5) Exit
-----------------------------------------
Enter your choice : 1
Enter the element to be inserted : 12


Menu
-----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Display
5) Exit
-----------------------------------------
Enter your choice : 2
Enter the element to be inserted : 36
```



```
Menu
-----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Display
5) Exit
-----------------------------------------
Enter your choice : 3
Enter the position to be inserted : 2
Enter the element to be inserted : 24
```



```
Menu
-----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Display
5) Exit
-----------------------------------------
Enter your choice : 3
Enter the position to be inserted : 48
Enter the element to be inserted : 12
Position not found
```



```
Menu
-----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Display
5) Exit
-----------------------------------------
Enter your choice : 4

The items are: 12 24 36
```

LAB 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 value;
struct node *next;
};


typedef struct node *NODE;

//For allocating memory space-

NODE getnode()
{
   NODE temp;
   temp=(NODE)malloc(sizeof(struct node));

   if(temp==NULL)
   {
       printf("Memory could not be allocated.");
   }

   return(temp);

}



//Inserting values at the beginning-

NODE insert_beg(NODE first,int item)
{
   NODE temp;
   temp=getnode();

   temp->value=item;
   temp->next=NULL;

   if(first==NULL)
       return temp;
   else
   {
       temp->next=first;
       first=temp;
       return first;
   }

}
```

```
//For deleting at the beginning-

NODE delete_beg(NODE first)
{
    NODE temp;
    // temp=getnode();

    if(first==NULL)
    {
        printf("There is nothing to delete");
        return NULL;
    }

    temp=first;
    temp=temp->next;

    printf("Item has been deleted:%d\n",first->value);
    free(first);

    return(temp);
}

//For deleting at the end-



NODE delete_end(NODE first)
```

```
{
    NODE prev,curr;

    if(first==NULL)
    {
        printf("Nothing to delete");
        return NULL;
    }

    // prev=getnode();
    // curr=getnode();

    prev=NULL;
    curr=first;

    while(curr->next!=NULL)
    {
        prev=curr;
        curr=curr->next;
    }

    prev->next=NULL;
    free(curr);
    return(first);
}



NODE del_specific(NODE first,int key)
```

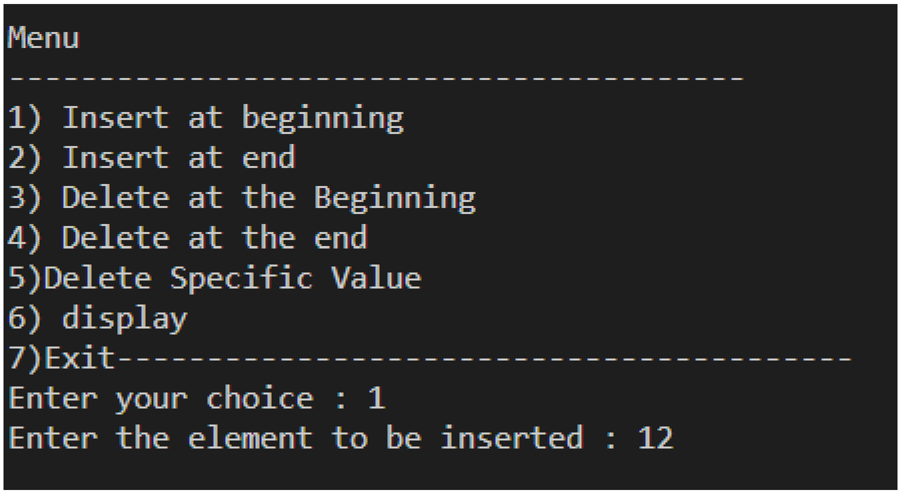
```
{
   NODE prev,curr;

   if(first==NULL)
   {   printf("Nothing to delete");
       return NULL;
   }

   curr=first;
   prev=NULL;

   if(curr->value==key)
   {
      printf("Item deleted:%d",curr->value);
      first=first->next;
      free(curr);
      return(first);
   }

   while((curr->value!=key)&&(curr!=NULL))
   {
      prev=curr;
      curr=curr->next;
   }

   if(curr->value==key)
   {
      prev->next=curr->next;
```

```
        printf("Item deleted:%d",curr->value);
        free(curr);
        return first;
    }

    if(curr==NULL)
    {
        printf("End of list reached but item wasnt found.");
        return first;
    }
}


//For displaying-

void display(NODE first)
{
    NODE temp;
    // temp=getnode();
    temp=first;

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

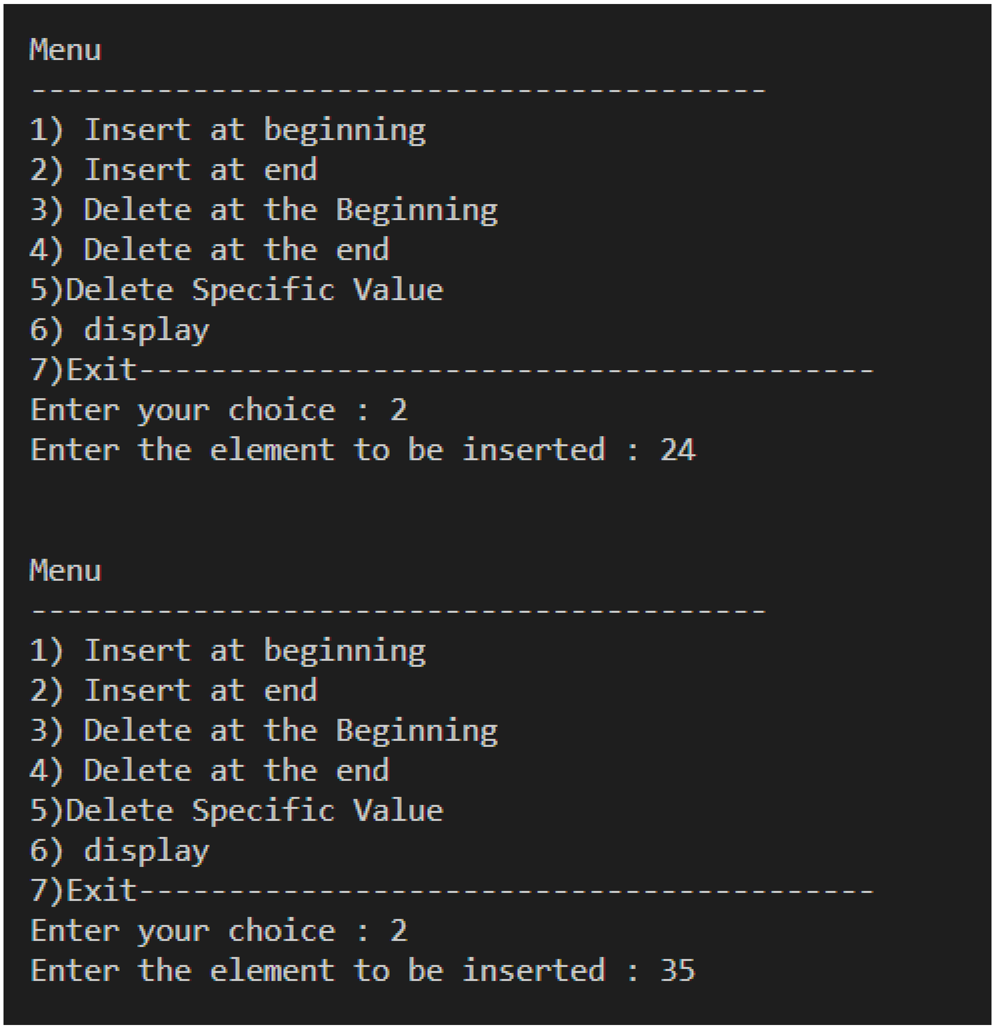
```
int main()

{

   int choice, item, x,pos;

   NODE first = NULL;

   while (1)

   {

      printf("\n\nMenu\n----------------------------------------\n1) Insert at beginning\n2) Insert at
end\n3) Display\n4) Delete at Beginning\n5) Delete at the end\n6) Delete a specific
Value\n7) Exit\n----------------------------------------\nEnter your choice : ");

      scanf("%d", &choice);

      switch (choice)

      {

      case 1:

         printf("Enter the element to be inserted : ");

         scanf("%d", &x);

         first = insert_beg(first,x);

         break;

      case 2:

         printf("Enter the element to be inserted : ");

         scanf("%d", &x);

         first = insert_end(first,x);

         break;

      case 3:printf("/nThe items are:");

         display(first);

         break;

      case 4:

         first = delete_beg(first);
```

```
            printf("Item has been removed from the first NODE");
            break;
        case 5:
            first = delete_end(first);
            printf("Item has been removed from the last node.");
            break;
        case 6:
            printf("Enter the value to be deleted-");
            scanf("%d",&x);
            first=del_specific(first,x);
            printf("\nItem has been removed from the list.");
            break;

        case 7:
             exit(0);
            break;
        default:
            printf("Please enter the correct value.");
        }
    }
    return 0;
}
```

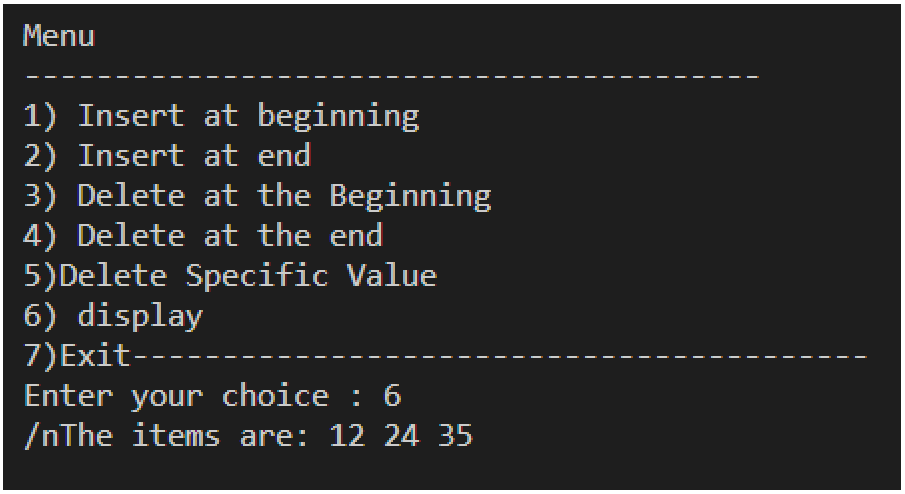
Output:



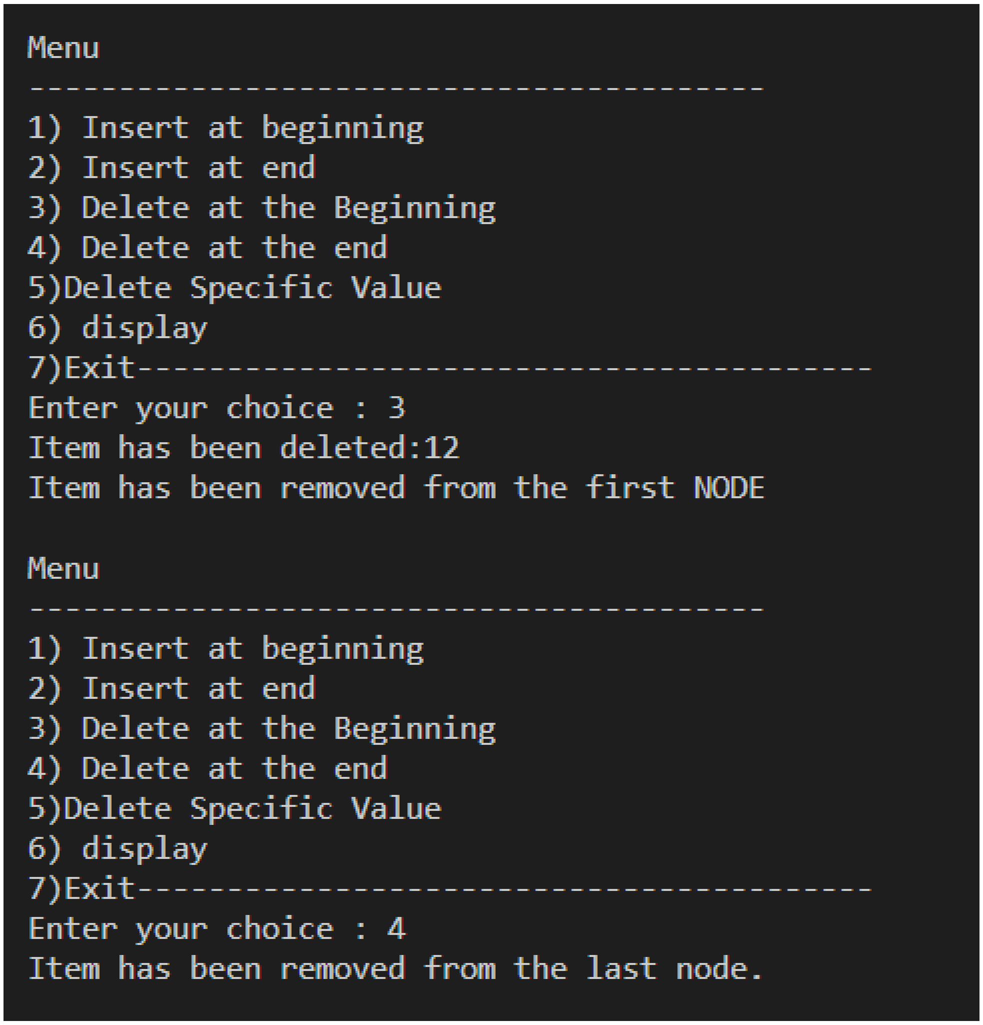
```
Menu
-----------------------------------------
1) Insert at beginning
2) Insert at end
3) Delete at the Beginning
4) Delete at the end
5)Delete Specific Value
6) display
7)Exit-----------------------------------------
Enter your choice : 1
Enter the element to be inserted : 12
```



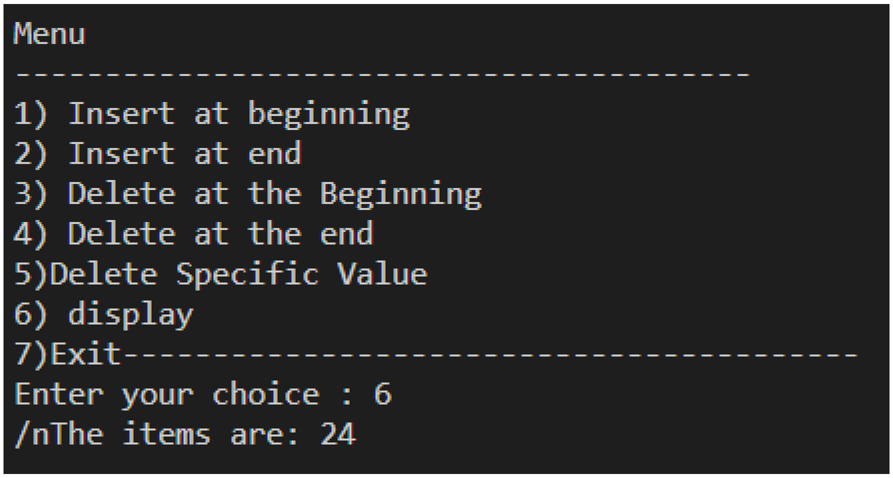
```
Menu
-----------------------------------------
1) Insert at beginning
2) Insert at end
3) Delete at the Beginning
4) Delete at the end
5)Delete Specific Value
6) display
7)Exit-----------------------------------------
Enter your choice : 2
Enter the element to be inserted : 24


Menu
-----------------------------------------
1) Insert at beginning
2) Insert at end
3) Delete at the Beginning
4) Delete at the end
5)Delete Specific Value
6) display
7)Exit-----------------------------------------
Enter your choice : 2
Enter the element to be inserted : 35
```



```
Menu
-----------------------------------------
1) Insert at beginning
2) Insert at end
3) Delete at the Beginning
4) Delete at the end
5)Delete Specific Value
6) display
7)Exit-----------------------------------------
Enter your choice : 6
/nThe items are: 12 24 35
```



```
Menu
-----------------------------------------
1) Insert at beginning
2) Insert at end
3) Delete at the Beginning
4) Delete at the end
5)Delete Specific Value
6) display
7)Exit-----------------------------------------
Enter your choice : 3
Item has been deleted:12
Item has been removed from the first NODE

Menu
-----------------------------------------
1) Insert at beginning
2) Insert at end
3) Delete at the Beginning
4) Delete at the end
5)Delete Specific Value
6) display
7)Exit-----------------------------------------
Enter your choice : 4
Item has been removed from the last node.
```



```
Menu
----------------------------------------
1) Insert at beginning
2) Insert at end
3) Delete at the Beginning
4) Delete at the end
5)Delete Specific Value
6) display
7)Exit----------------------------------------
Enter your choice : 6
/nThe items are: 24
```

LAB PROGRAM 7:

PROGRAM 7:

WAP 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 <stdlib.h>

struct NODE
{
   int value;
   struct NODE *next;
};
```

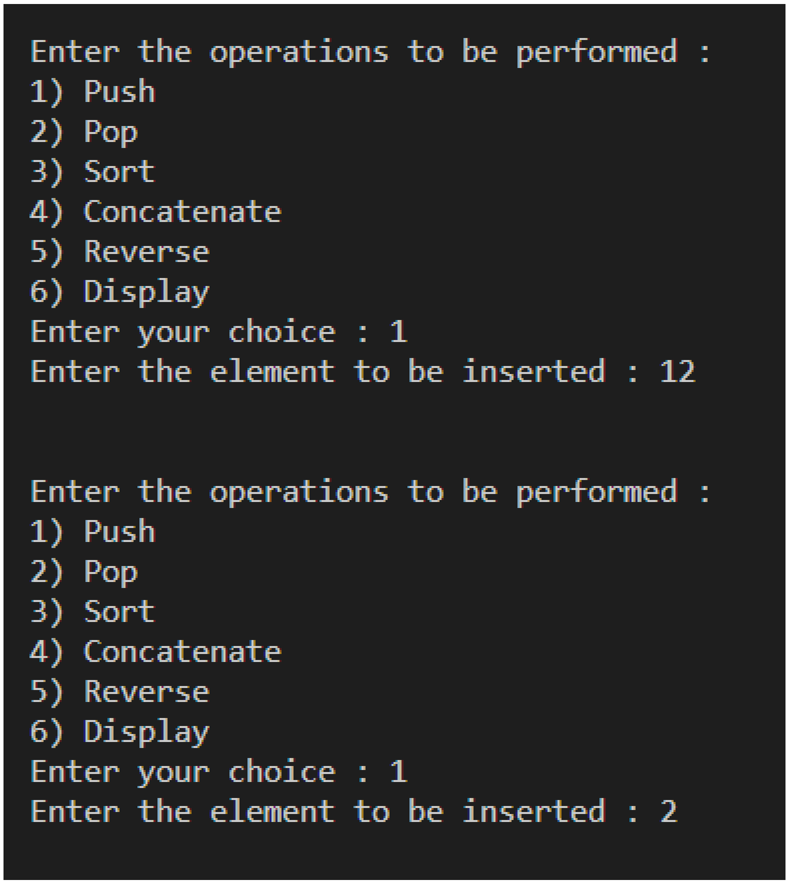
```
typedef struct NODE *node;

node insert_at_beginning(int item, node first)
{
    node temp = (node)malloc(sizeof(struct NODE));
    if (temp == NULL)
    {
        printf("\nMemory not allocated!");
    }
    (temp->value) = item;
    (temp->next) = NULL;
    if (first == NULL)
    {
        return temp;
    }
    else
    {
        temp->next = first;
        first = temp;
        return first;
    }
}

node delete_at_the_beginning(node first)
{
    if (first == NULL)
    {
        printf("Cannot delete, the Linked List is empty");
        return NULL;
    }
    else
    {
        node temp;
        temp = first;
        first = (first->next);
        free(temp);
        return first;
    }
}

node sort(node first)
{
    int temp;
    node curr = first;
    if (first == NULL)
    {
        printf("Linked list is empty!");
        return NULL;
    }
    else
    {
        while (curr->next != NULL)
        {
            node check = curr->next;
            while (check != NULL)
            {
                if (curr->value > check->value)
                {
                    temp = curr->value;
                    curr->value = check->value;
                    check->value = temp;
                }
                check = check->next;
            }
            curr = curr->next;
        }
        return first;
    }
}

node concatenate(node f1, node f2)
{
    if (f1 == NULL && f2 == NULL)
    {
        printf("The linked lists are empty!");
        return NULL;
    }
    else if (f1 != NULL && f2 == NULL)
        return f1;
    else if (f1 == NULL && f2 != NULL)
        return f2;
    else
    {
        node last = f1;
        while (last->next != NULL)
        {
            last = last->next;
        }
        last->next = f2;
        return f1;
    }
}

node reverse(node first)
{
    if (first == NULL)
    {
        printf("The linked lists are empty!");
        return NULL;
    }
    else
    {
        node rev = NULL;
        while (first != NULL)
        {
            node Next = first->next;
            first->next = rev;
            rev = first;
            first = Next;
        }
        return rev;
    }
```

```
}

void display(node first)
{
    node temp;
    temp = first;
    if (temp == NULL)
    {
        printf("The Linked list is empty!");
    }
    else
    {
        printf("The elements in the node are :  ");
        while (temp != NULL)
        {
            printf("%d  ", (temp->value));
            temp = (temp->next);
        }
    }
}

int main()
{
    int choice, n, i, val, x;
    node first = NULL, f1 = NULL, f2 = NULL;
    while (1)
    {
```

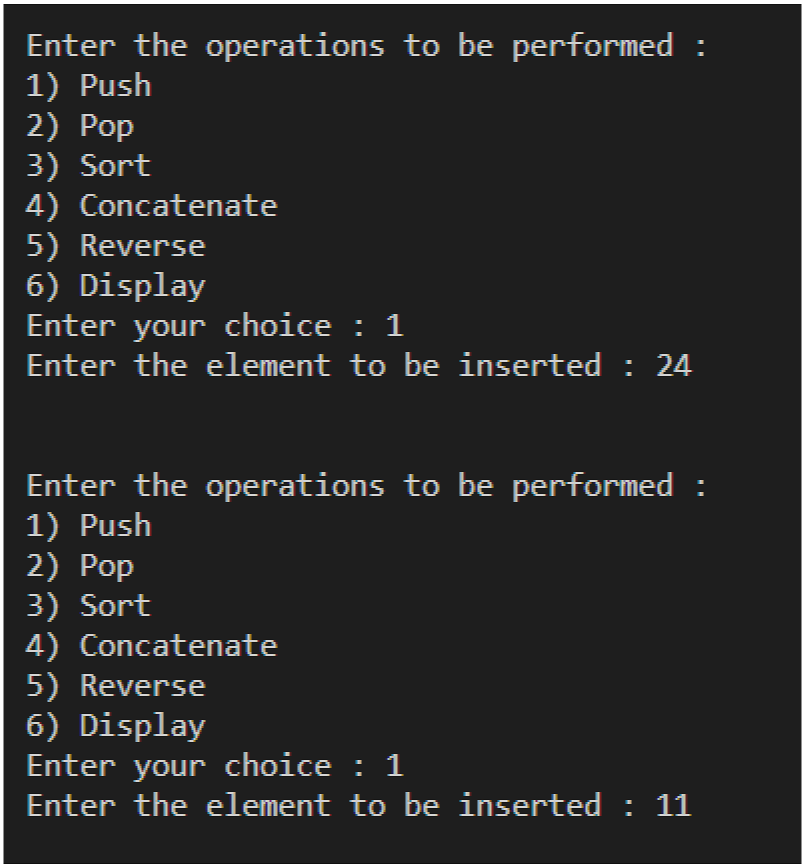
```
    printf("\n\nEnter the operations to be performed :\n1) Push\n2) Pop\n3) Sort\n4)
Concatenate\n5) Reverse\n6) Display\nEnter your choice : ");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
      printf("Enter the element to be inserted : ");
      scanf("%d", &x);
      first = insert_at_beginning(x, first);
      break;
    case 2:
      first = delete_at_the_beginning(first);
      break;
    case 3:
      first = sort(first);
      break;
    case 4:

      printf("Enter the number of fields for linked list 1 : ");
      scanf("%d", &n);
      printf("Enter %d entries : ", n);
      for (i = 0; i < n; i++)
      {
        scanf("%d", &val);
        f1 = insert_at_beginning(val, f1);
      }
      printf("Enter the number of fields for linked list 2 : ");
      scanf("%d", &n);
```

```
            printf("Enter %d entries : ", n);
            for (i = 0; i < n; i++)
            {
                scanf("%d", &val);
                f2 = insert_at_beginning(val, f2);
            }
            printf("The concatenated linked list is : ");
            f1 = concatenate(f1, f2);
            display(f1);
            break;
        case 5:
            first = reverse(first);
            break;
        case 6:
            display(first);
            break;
        default:
            exit(0);
        }
    }
    return 0;
}
```

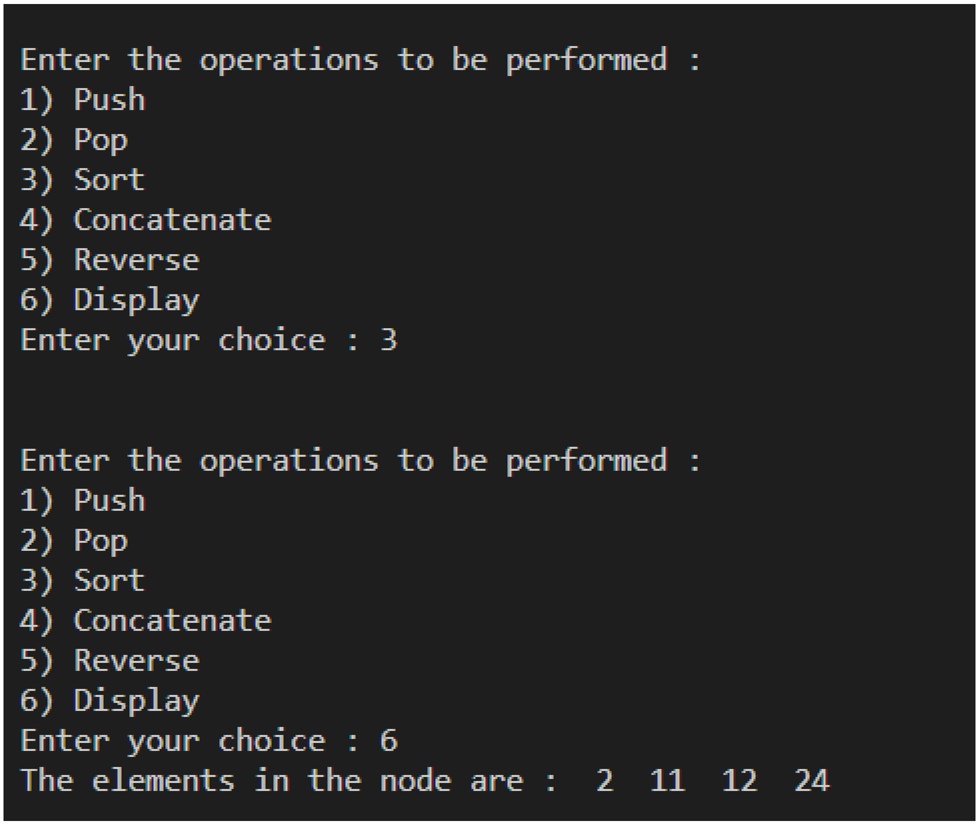
Output:



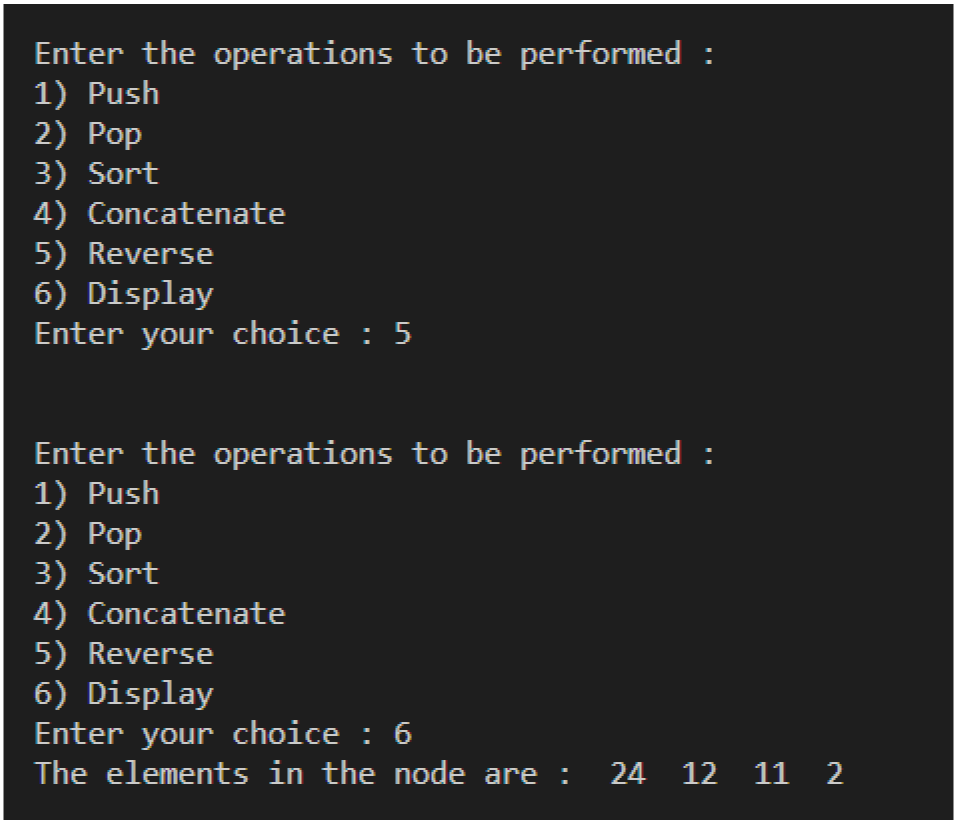
```
Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 1
Enter the element to be inserted : 12


Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 1
Enter the element to be inserted : 2
```



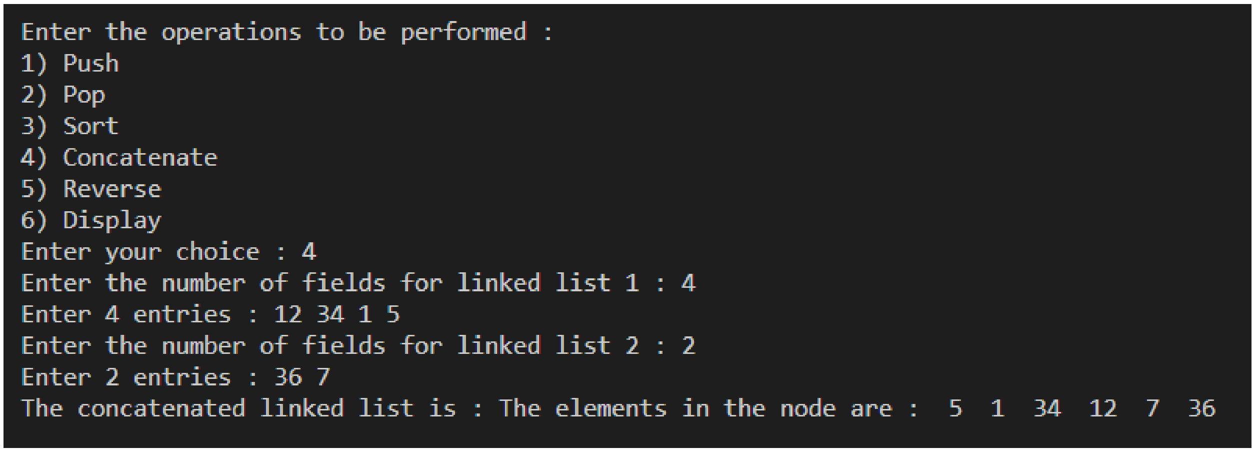
```
Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 1
Enter the element to be inserted : 24


Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 1
Enter the element to be inserted : 11
```



```
Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 3


Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 6
The elements in the node are :  2  11  12  24
```



```
Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 5


Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 6
The elements in the node are :  24  12  11  2
```



```
Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 4
Enter the number of fields for linked list 1 : 4
Enter 4 entries : 12 34 1 5
Enter the number of fields for linked list 2 : 2
Enter 2 entries : 36 7
The concatenated linked list is : The elements in the node are :  5  1  34  12  7  36
```

LAB PROGRAM 8:

Write a program to implement Stacks and Queues using a linked list.

Code for Implementation of Stack:

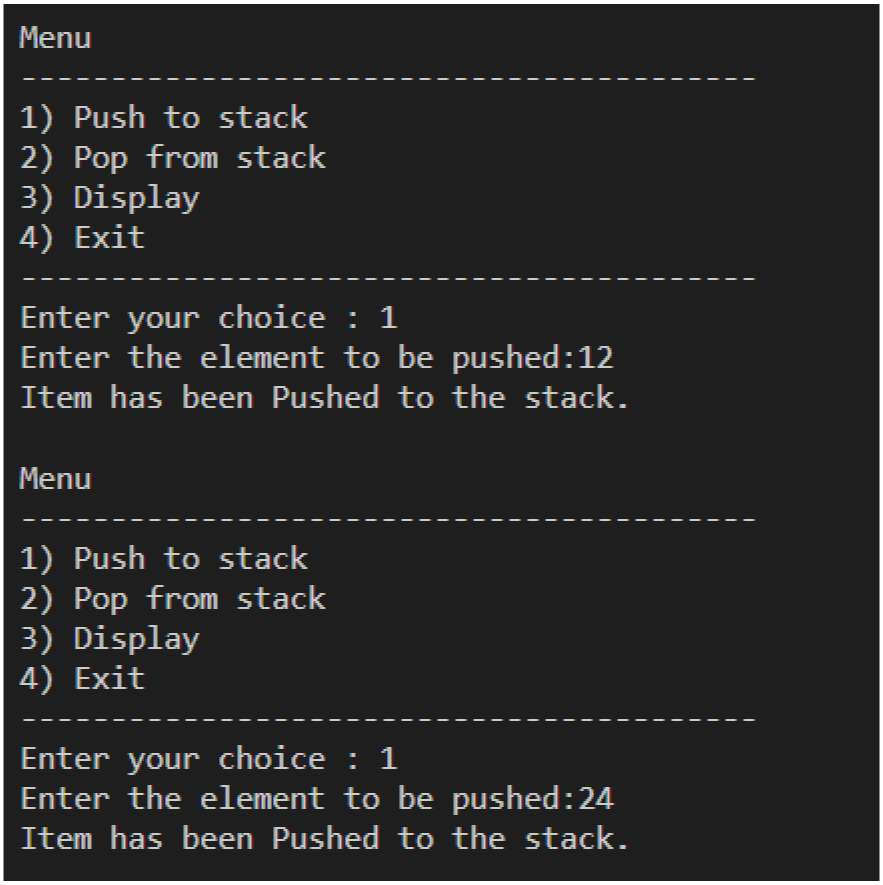
```
#include<stdio.h>
#include<stdlib.h>

struct node{
int value;
struct node *next;
};


typedef struct node *NODE;

//For allocating memory space-

NODE getnode()
{
   NODE temp;
   temp=(NODE)malloc(sizeof(struct node));

   if(temp==NULL)
   {
      printf("Memory could not be allocated.");
   }
```

```
    return(temp);

}


//For inserting at the end-

NODE insert_end(NODE first,int item)
{
    NODE New,last;
    New=getnode();
    New->value=item;
    New->next=NULL;

    if(first==NULL)
        return New;

    if(first->next==NULL)
    {
    first->next=New;
    return first;
    }

    // last=getnode();
    last=first;

    while(last->next!=NULL)
        last=last->next;
    last->next=New;



    return first;

}



//For deleting at the end-


NODE delete_end(NODE first)
{
    NODE prev,curr;

    if(first==NULL)
    {
        printf("Nothing to delete");
        return NULL;
    }

    // prev=getnode();
    // curr=getnode();

    prev=NULL;
    curr=first;

    while(curr->next!=NULL)
    {
        prev=curr;
        curr=curr->next;
    }

    prev->next=NULL;
    printf("Deleted Item:%d",curr->value);
    free(curr);
    return(first);
}



void display(NODE first)
{
    NODE temp;
    // temp=getnode();
    temp=first;

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

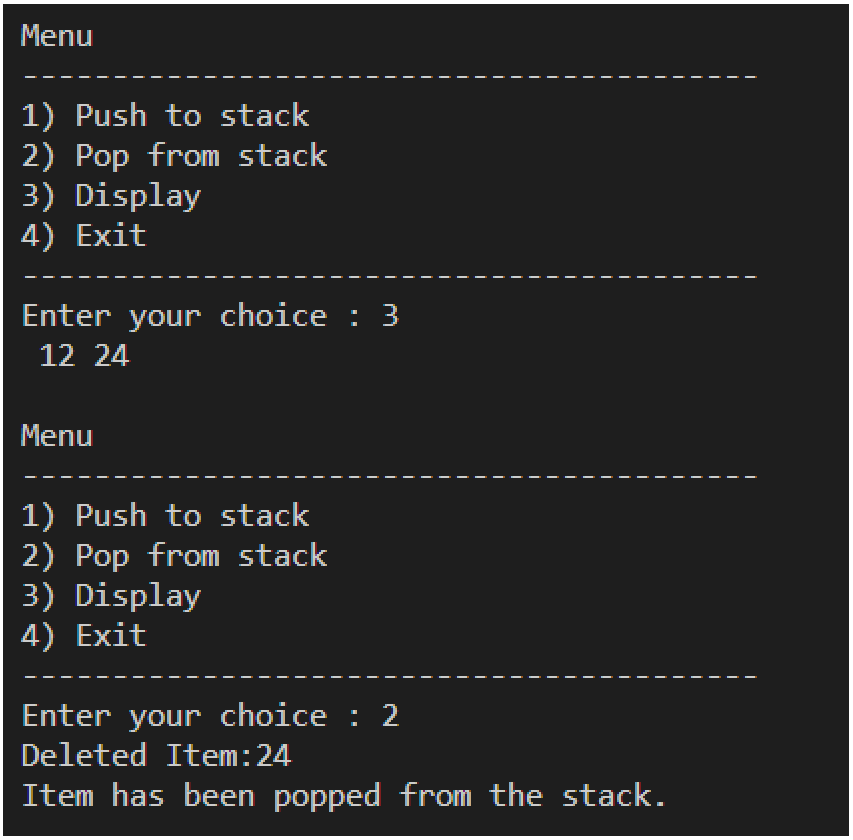
```
}



int main(){


 int choice, item, x,pos;
    NODE first = NULL;
    while (1)
    {
        printf("\n\nMenu\n----------------------------------------\n1) Push to stack\n2) Pop from 
stack\n3) Display\n4) Exit\n----------------------------------------\nEnter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1: printf("Enter the element to be pushed:");
                scanf("%d",&item);
                first=insert_end(first,item);
                printf("Item has been Pushed to the stack.");
                break;
        case 2: first=delete_end(first);
                printf("\nItem has been popped from the stack.");
                break;
        case 3: display(first);
                break;
        case 4: exit(0);
        default:
        printf("Please Enter the correct value.");
```

```
            break;
        }
    }

    return 0;
}
```

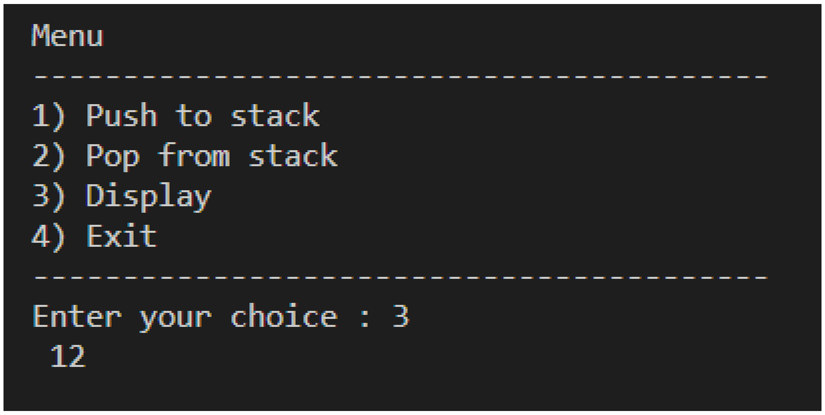
Output:



```
Menu
-----------------------------------------
1) Push to stack
2) Pop from stack
3) Display
4) Exit
-----------------------------------------
Enter your choice : 1
Enter the element to be pushed:12
Item has been Pushed to the stack.

Menu
-----------------------------------------
1) Push to stack
2) Pop from stack
3) Display
4) Exit
-----------------------------------------
Enter your choice : 1
Enter the element to be pushed:24
Item has been Pushed to the stack.
```



```
Menu
----------------------------------------
1) Push to stack
2) Pop from stack
3) Display
4) Exit
----------------------------------------
Enter your choice : 3
 12 24

Menu
----------------------------------------
1) Push to stack
2) Pop from stack
3) Display
4) Exit
----------------------------------------
Enter your choice : 2
Deleted Item:24
Item has been popped from the stack.
```



```
Menu
----------------------------------------
1) Push to stack
2) Pop from stack
3) Display
4) Exit
----------------------------------------
Enter your choice : 3
 12
```

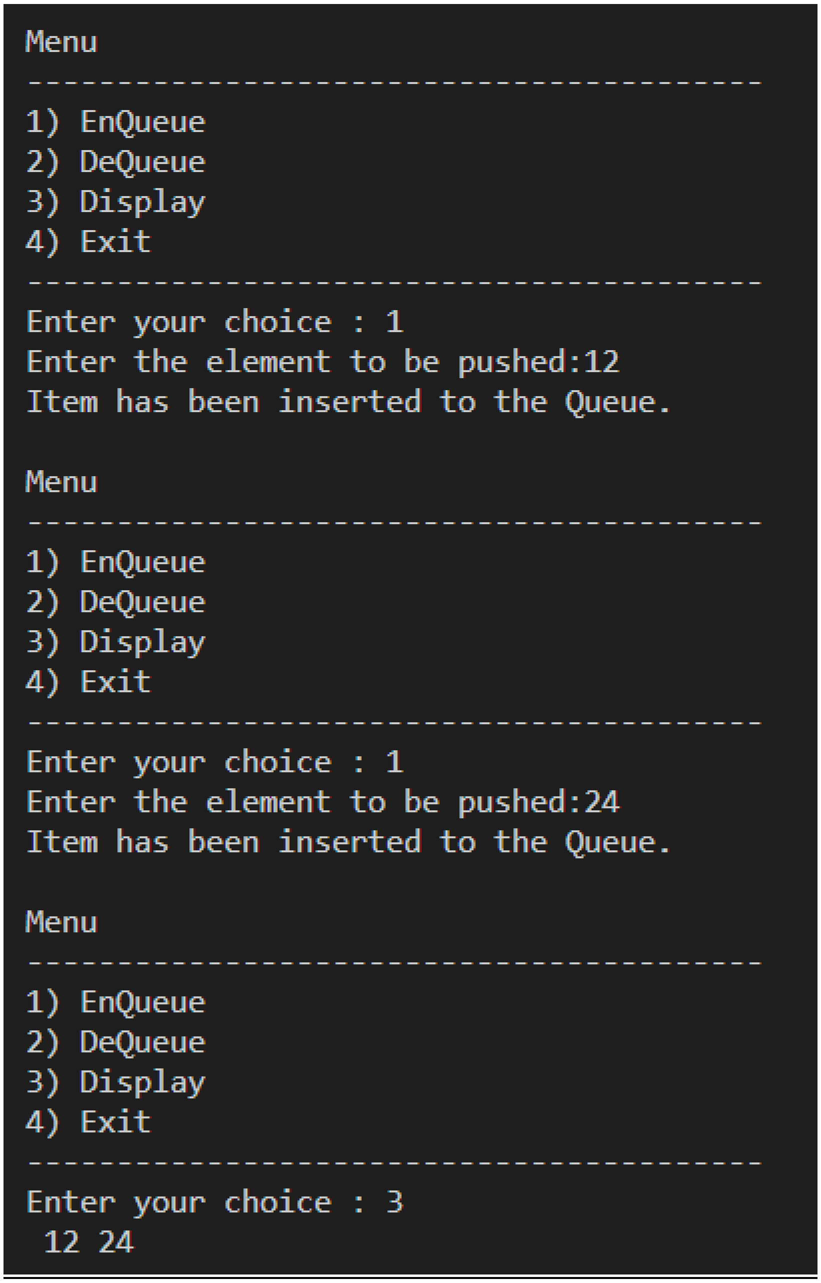
Code for Implementation of Queue:

```
#include<stdio.h>

#include<stdlib.h>


struct node{

int value;
```

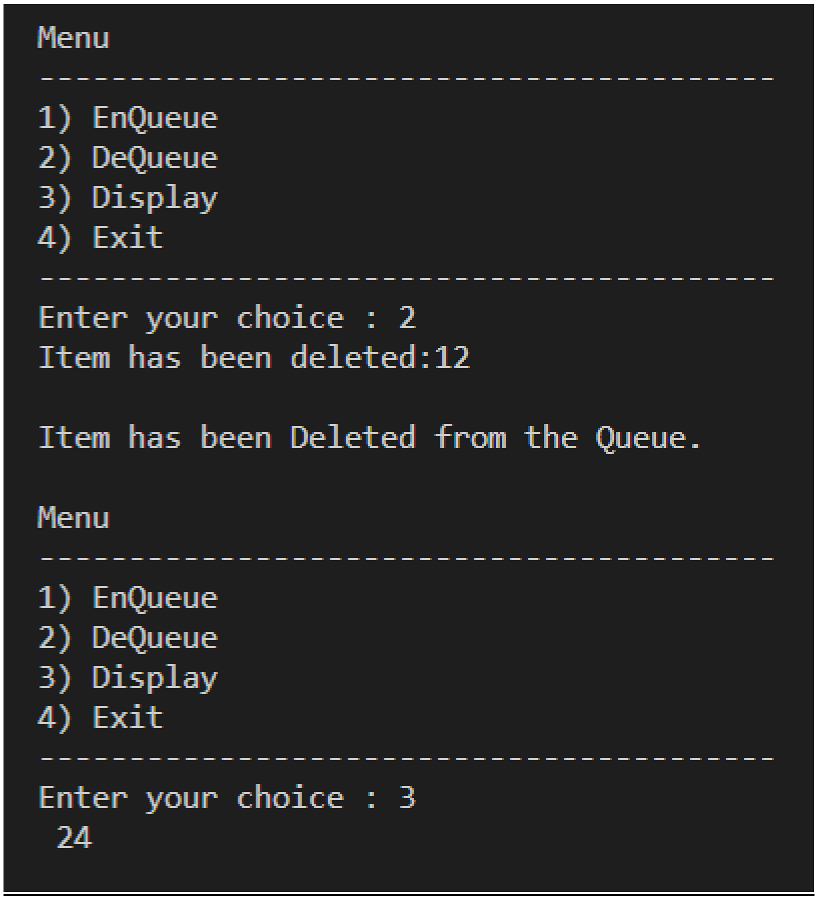
```
struct node *next;
};


typedef struct node *NODE;

//For allocating memory space-

NODE getnode()
{
    NODE temp;
    temp=(NODE)malloc(sizeof(struct node));

    if(temp==NULL)
    {
        printf("Memory could not be allocated.");
    }

    return(temp);

}


//For inserting at the end-

NODE insert_end(NODE first,int item)
{
    NODE New,last;
    New=getnode();
    New->value=item;
    New->next=NULL;

    if(first==NULL)
        return New;

    if(first->next==NULL)
    {
    first->next=New;
    return first;
    }

    // last=getnode();
    last=first;

    while(last->next!=NULL)
        last=last->next;

    last->next=New;



    return first;

}
```

```
//For deleting at the beginning-

NODE delete_beg(NODE first)
{
    NODE temp;
    // temp=getnode();

    if(first==NULL)
    {
        printf("There is nothing to delete");
        return NULL;
    }

    temp=first;
    temp=temp->next;

    printf("Item has been deleted:%d\n",first->value);
    free(first);

    return(temp);
}



//For displaying-

void display(NODE first)
{
    NODE temp;
    // temp=getnode();
    temp=first;

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


int main(){

 int choice, item;
    NODE first = NULL;
    while (1)
    {
        printf("\n\nMenu\n--------------------------------------\n1) EnQueue\n2) DeQueue\n3)
Display\n4) Exit\n--------------------------------------\nEnter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1: printf("Enter the element to be pushed:");
                scanf("%d",&item);
                first=insert_end(first,item);
                printf("Item has been inserted to the Queue.");
                break;
        case 2: first=delete_beg(first);

               printf("\nItem has been Deleted from the Queue.");
               break;
        case 3: display(first);
               break;
        case 4: exit(0);
        default:
        printf("Please Enter the correct value.");
           break;
        }
}

return 0;
}
```

Output:



```
Menu
-----------------------------------------
1) EnQueue
2) DeQueue
3) Display
4) Exit
-----------------------------------------
Enter your choice : 1
Enter the element to be pushed:12
Item has been inserted to the Queue.

Menu
-----------------------------------------
1) EnQueue
2) DeQueue
3) Display
4) Exit
-----------------------------------------
Enter your choice : 1
Enter the element to be pushed:24
Item has been inserted to the Queue.

Menu
-----------------------------------------
1) EnQueue
2) DeQueue
3) Display
4) Exit
-----------------------------------------
Enter your choice : 3
 12 24
```



```
Menu
-----------------------------------------
1) EnQueue
2) DeQueue
3) Display
4) Exit
-----------------------------------------
Enter your choice : 2
Item has been deleted:12

Item has been Deleted from the Queue.

Menu
-----------------------------------------
1) EnQueue
2) DeQueue
3) Display
4) Exit
-----------------------------------------
Enter your choice : 3
 24
```

LAB PROGRAM 9:

WAP Implement doubly link list with primitive operations

a) Create a doubly linked list.

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

c) Delete the node based on a specific value

d) Display the contents of the list

Code:

```
#include<stdio.h>

#include<stdlib.h>
```

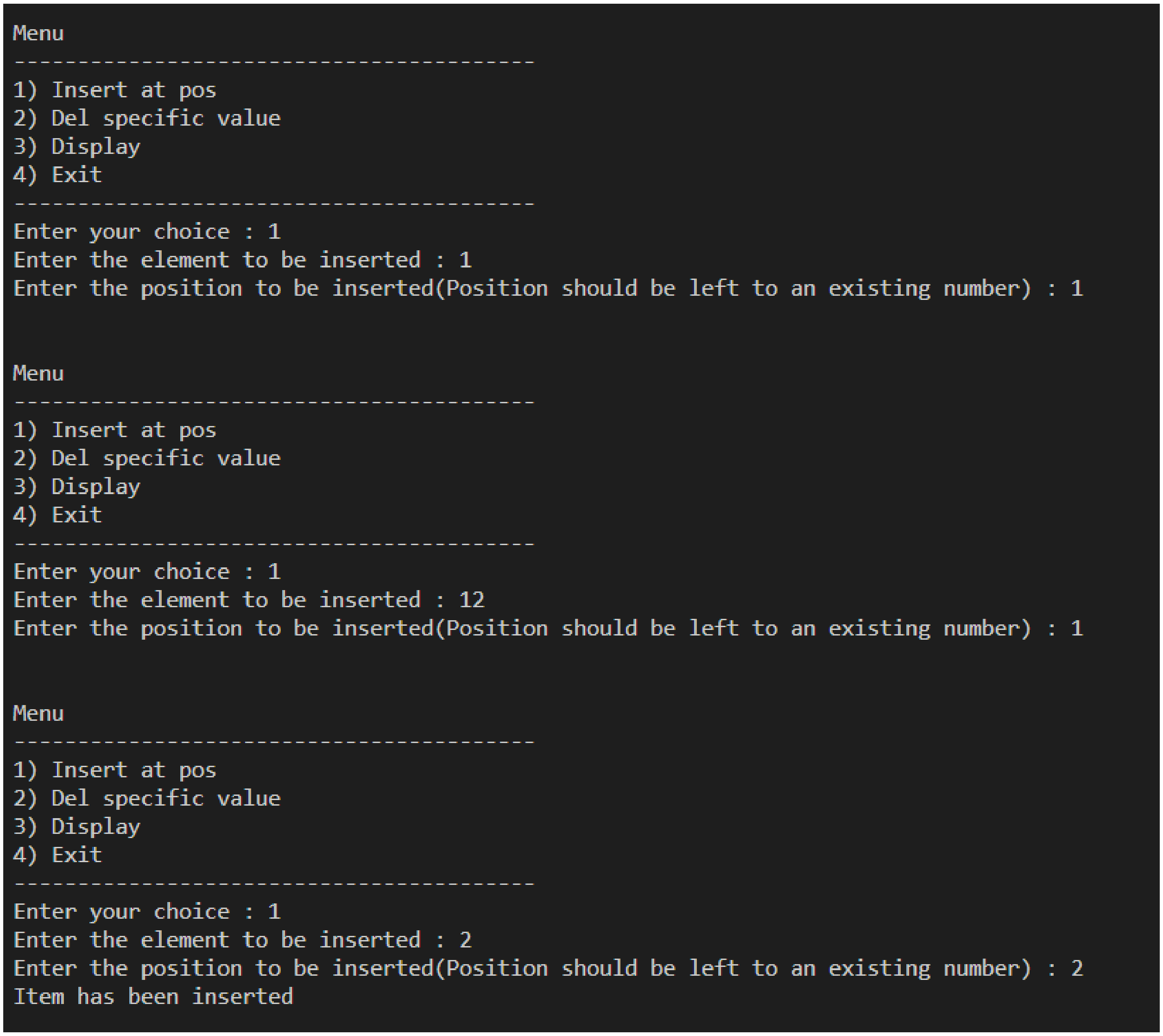
```
struct node{
int value;
struct node *next;
struct node *prev;
};


typedef struct node *NODE;

//For allocating memory space-

NODE getnode()
{
    NODE temp;
    temp=(NODE)malloc(sizeof(struct node));

    if(temp==NULL)
    {
        printf("Memory could not be allocated.");
    }

    return(temp);

}


NODE insert_left(NODE first,int pos,int item){
```

```
// printf("HI");

int count=1;
NODE New,curr;
New = getnode();
New->value=item;
New->next=NULL;
New->prev=NULL;

if(first==NULL){
    // printf("ONe more hi");
    return New;
}

if(pos==1){
    New->next=first;
    first->prev=New;
    first = New;
    return first;
}

// printf("Other hi");

curr = first;
while(count!=pos && curr->next!=NULL){
    curr=curr->next;
    count++;
}
```

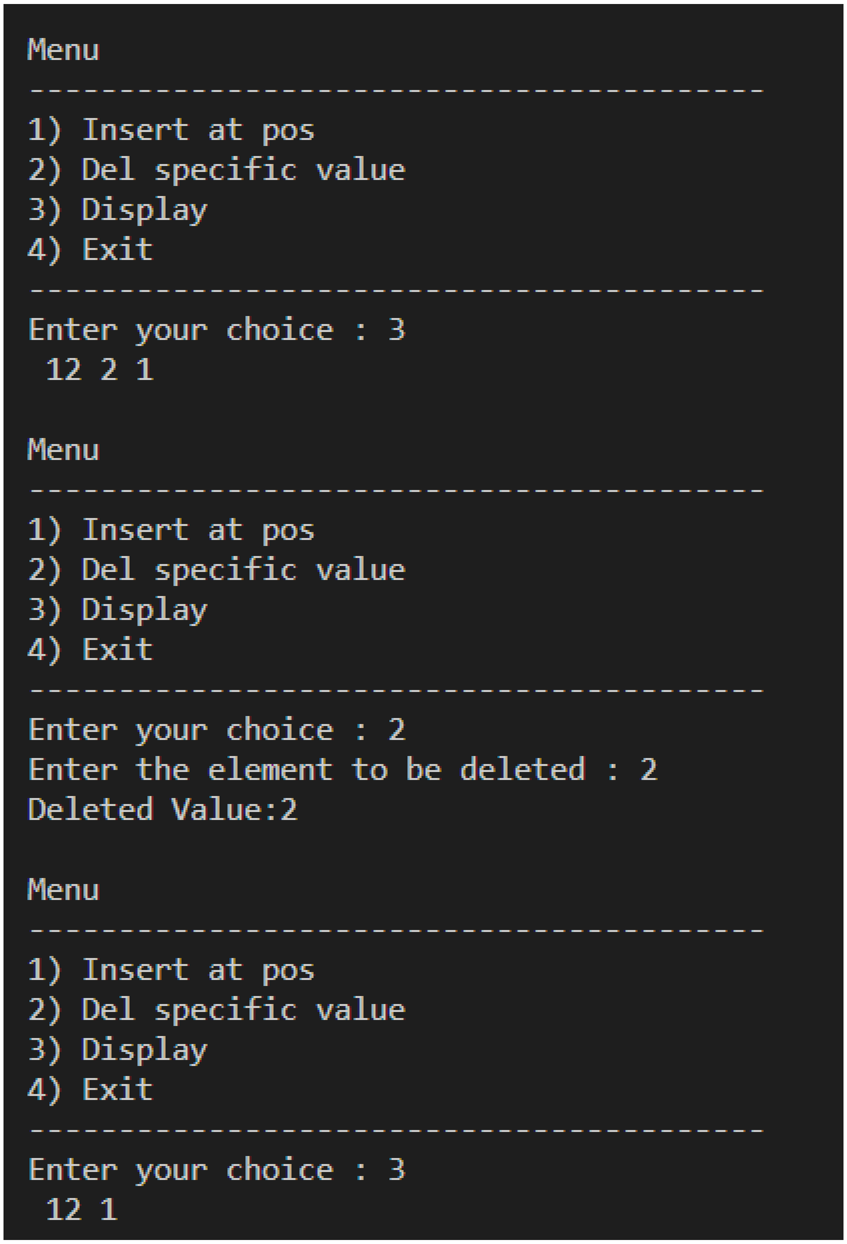
```
    if(count==pos){
        New->next = curr;
        New->prev = curr->prev;
        (curr->prev)->next = New;
        curr->prev = New;
        printf("Item has been inserted");
        return first;
    }

    printf("Position couldnt be found");
    return first;


}




NODE del_spec(NODE first,int key){
    // int count=1;
    NODE temp,curr;

    if(first==NULL){
       printf("Nothing to delete :(");
       return NULL;
    }
```

```
if(key==first->value){
    first=first->next;
    return first;
}

curr = first;

while(curr->next!=NULL && curr->value!=key){
    curr = curr->next;
}

if(curr->next==NULL && curr->value==key){
    (curr->prev)->next = curr->next;
    printf("Deleted Value:%d",curr->value);
    free(curr);
    return first;
}

if(curr->value==key){
    (curr->prev)->next = curr->next;
    (curr->next)->prev = curr->prev;

    printf("Deleted Value:%d",curr->value);
    free(curr);
    return first;
}

printf("Couldn't find Value :(");
    return first;

}



void display(NODE first)
{
    NODE temp;
    // temp=getnode();
    temp=first;

    if(first == NULL){
        printf("Whoops List is empty!");
    }

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




int main()
{
    int choice, item, x;
    NODE first = NULL;
    while (1)
    {
        printf("\n\nMenu\n-------------------------------------\n1) Insert at pos\n2) Del specific value\n3) Display\n4) Exit\n-------------------------------------\nEnter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
                printf("Enter the element to be inserted : ");
                scanf("%d", &item);
                printf("Enter the position to be inserted : ");
                scanf("%d", &x);
                first = insert_left(first,x,item);
                break;
        case 2: printf("Enter the element to be deleted : ");
                scanf("%d", &item);
                first = del_spec(first,item);
                break;
        case 3:
                display(first);
                break;
        case 4: printf("Exiting...");
                exit(0);
                break;
        default:printf("Please enter correct choice :(");
                break;
        }
    }
    return 0;
}
```

Output:



```
Menu
-----------------------------------------
1) Insert at pos
2) Del specific value
3) Display
4) Exit
-----------------------------------------
Enter your choice : 1
Enter the element to be inserted : 1
Enter the position to be inserted(Position should be left to an existing number) : 1


Menu
-----------------------------------------
1) Insert at pos
2) Del specific value
3) Display
4) Exit
-----------------------------------------
Enter your choice : 1
Enter the element to be inserted : 12
Enter the position to be inserted(Position should be left to an existing number) : 1


Menu
-----------------------------------------
1) Insert at pos
2) Del specific value
3) Display
4) Exit
-----------------------------------------
Enter your choice : 1
Enter the element to be inserted : 2
Enter the position to be inserted(Position should be left to an existing number) : 2
Item has been inserted
```



```
Menu
-----------------------------------------
1) Insert at pos
2) Del specific value
3) Display
4) Exit
-----------------------------------------
Enter your choice : 3
 12 2 1

Menu
-----------------------------------------
1) Insert at pos
2) Del specific value
3) Display
4) Exit
-----------------------------------------
Enter your choice : 2
Enter the element to be deleted : 2
Deleted Value:2

Menu
-----------------------------------------
1) Insert at pos
2) Del specific value
3) Display
4) Exit
-----------------------------------------
Enter your choice : 3
 12 1
```

LAB 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 postorder

c) To display the elements in the tree.

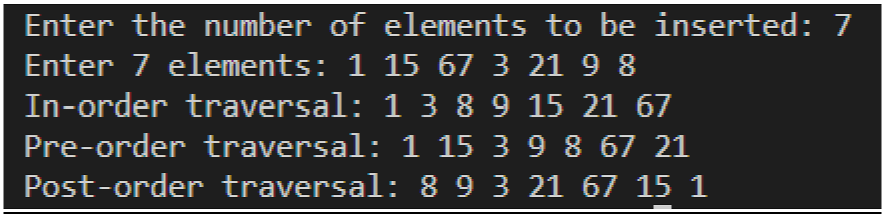
Code:

```
#include <stdio.h>

#include <stdlib.h>


struct node

{

   int data;

   struct node *left;

   struct node *right;

};


struct node *insert(struct node *node, int data)

{

   if (node == NULL)

   {

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

      temp->data = data;

      temp->left = temp->right = NULL;

      return temp;

   }
```

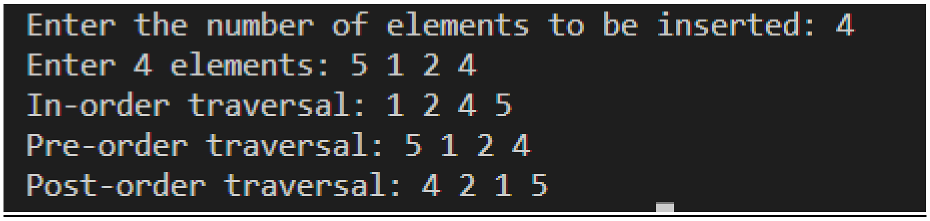
```
    if (data < node->data)
        node->left = insert(node->left, data);
    else if (data > node->data)
        node->right = insert(node->right, data);

    return node;
}

void inorder(struct node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}

void preorder(struct node *root)
{
    if (root != NULL)
    {
        printf("%d ", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}
```

```
void postorder(struct node *root)
{
    if (root != NULL)
    {
        postorder(root->left);
        postorder(root->right);
        printf("%d ", root->data);
    }
}

int main()
{
    struct node *root = NULL;
    int n, i, element;

    printf("Enter the number of elements to be inserted: ");
    scanf("%d", &n);
    printf("Enter %d elements: ", n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &element);
        root = insert(root, element);
    }

    printf("In-order traversal: ");
    inorder(root);
    printf("\nPre-order traversal: ");
    preorder(root);
    printf("\nPost-order traversal: ");
    postorder(root);

    return 0;
}
```

Output:



```
Enter the number of elements to be inserted: 7
Enter 7 elements: 1 15 67 3 21 9 8
In-order traversal: 1 3 8 9 15 21 67
Pre-order traversal: 1 15 3 9 8 67 21
Post-order traversal: 8 9 3 21 67 15 1
```



```
Enter the number of elements to be inserted: 4
Enter 4 elements: 5 1 2 4
In-order traversal: 1 2 4 5
Pre-order traversal: 5 1 2 4
Post-order traversal: 4 2 1 5
```