

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



DATA STRUCTURE LAB RECORD

Submitted by

Shashank Sharma (1BM19CS147)

Under the Guidance of

**Prof. Lohith J.J.
Assistant Professor, BMSCE**

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
Sep-2020 to Jan-2021

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 RECORD carried out by **Shashank Sharma(1BM19CS147)** who is the bonafide students 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 Visveswaraiiah Technological University, Belgaum during the year 2020-2021. The lab report has been approved as it satisfies the academic requirements in respect of **DATA STRUCTURE LAB RECORD (19CS3PCDST)** work prescribed for the said degree.

Signature of the Guide
Prof. Lohith J.J.
Assistant Professor
BMSCE, Bengaluru

Signature of the HOD
Dr. Umadevi V
Associate Prof.& Head, Dept. of CSE
BMSCE, Bengaluru

External Viva

Name of the Examiner

Signature with date

1. _____

2. _____

Data Structures Lab Record

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

```
#include<stdio.h>
#include<stdlib.h>
int size;
int arr[25];
int top=-1;
int item;

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

int main()
{
    int item_del;
    int ch;
    printf("Enter size of stack\n");
    scanf("%d",&size);
    for(;;)
    {
        printf("\n1.Push\n2.Pop\n3.Display\n0.Exit\n");fflush(stdin);
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:    push();
                      break;
            case 2: item_del=pop();
                      if(item_del==-1)
                          printf("Stack is Empty\n");
                      else
                          printf("Item Deleted: %d\n",item_del);
        }
    }
}
```

```

                break;
            case 3: display();
                break;
            case 0: exit(0);
                break;
            default: printf("Invalid choice\n");
        }
    }
    return 0;
}

void push()
{
    if(top==size-1)
    {
        printf("Stack is filled\n");
        return;
    }
    printf("Enter Item to be inserted in Stack\n"); fflush(stdin);
    scanf("%d",&item);
    top++;
    arr[top]=item;
}

int pop()
{
    if(top== -1)
    {
        return -1;
    }
    else
    {
        return arr[top--];
    }
}

void display()
{
    int i;

```

```
if(top==-1)
{
    printf("Stack is Empty\n");
}
else
for(i=0;i<=top;i++)
{
    printf("Element %d: %d\n",i+1,arr[i]);
}
}
```

```

Enter size of stack
5

1.Push
2.Pop
3.Display
0.Exit
1
Enter Item to be inserted in Stack
1

1.Push
2.Pop
3.Display
0.Exit
1
Enter Item to be inserted in Stack
2

1.Push
2.Pop
3.Display
0.Exit
1
Enter Item to be inserted in Stack
3

1.Push
2.Pop
3.Display
0.Exit
1
Enter Item to be inserted in Stack
4

1.Push
2.Pop
3.Display
0.Exit
1
Enter Item to be inserted in Stack
5

1.Push
2.Pop
3.Display
0.Exit
1
Stack is filled(Overflow)

1.Push
2.Pop
3.Display
0.Exit
3
Element 1: 1
Element 2: 2
Element 3: 3
Element 4: 4
Element 5: 5

1.Push
2.Pop
3.Display
0.Exit
2
Item Deleted: 5

1.Push
2.Pop
3.Display
0.Exit
2

```

```
2
Item Deleted: 4
1.Push
2.Pop
3.Display
0.Exit
2
Item Deleted: 3
1.Push
2.Pop
3.Display
0.Exit
2
Item Deleted: 2
1.Push
2.Pop
3.Display
0.Exit
2
Item Deleted: 1
1.Push
2.Pop
3.Display
0.Exit
2
Stack is Empty<Underflow>
1.Push
2.Pop
3.Display
0.Exit
2
Stack is Empty<Underflow>
1.Push
2.Pop
3.Display
0.Exit
3
Stack is Empty
1.Push
2.Pop
3.Display
0.Exit
0
Exiting
```

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)

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
int F(char symbol)
{
    switch(symbol)
    {
        case '+':
        case '-': return 2;
        case '*':
        case '/': return 4;
        case '^':
        case '$': return 5;
        case '(': return 0;
        case '#': return -1;
        default : return 8;
    }
}
int G(char symbol)
{
    switch(symbol)
    {
        case '+':
        case '-': return 1;
        case '*':
        case '/': return 3;
        case '^':
        case '$': return 6;
        case '(': return 9;
```



```

        case ')': return 0;
        default : return 7;
    }
}
void infix_postfix(char infix[],char postfix[])
{
    int top,j,i;
    char s[30];
    char symbol;
    top=-1;
    s[++top]='#';
    j=0;
    for(i=0;i<strlen(infix);i++)
    {
        symbol=infix[i];
        while(F(s[top])>G(symbol))
        {

            postfix[j]=s[top--];
            j++;
        }
        if(F(s[top])!=G(symbol))
            s[++top]=symbol;
        else
            top--;
    }
    while(s[top]!='#')
    {
        postfix[j++]=s[top--];
    }
    postfix[j]='\0';
}
int main()
{
    char infix[20],postfix[20];
    int k,a=0,b=0,flag=0;
    printf("Enter the valid infix expression:\n");

```

```

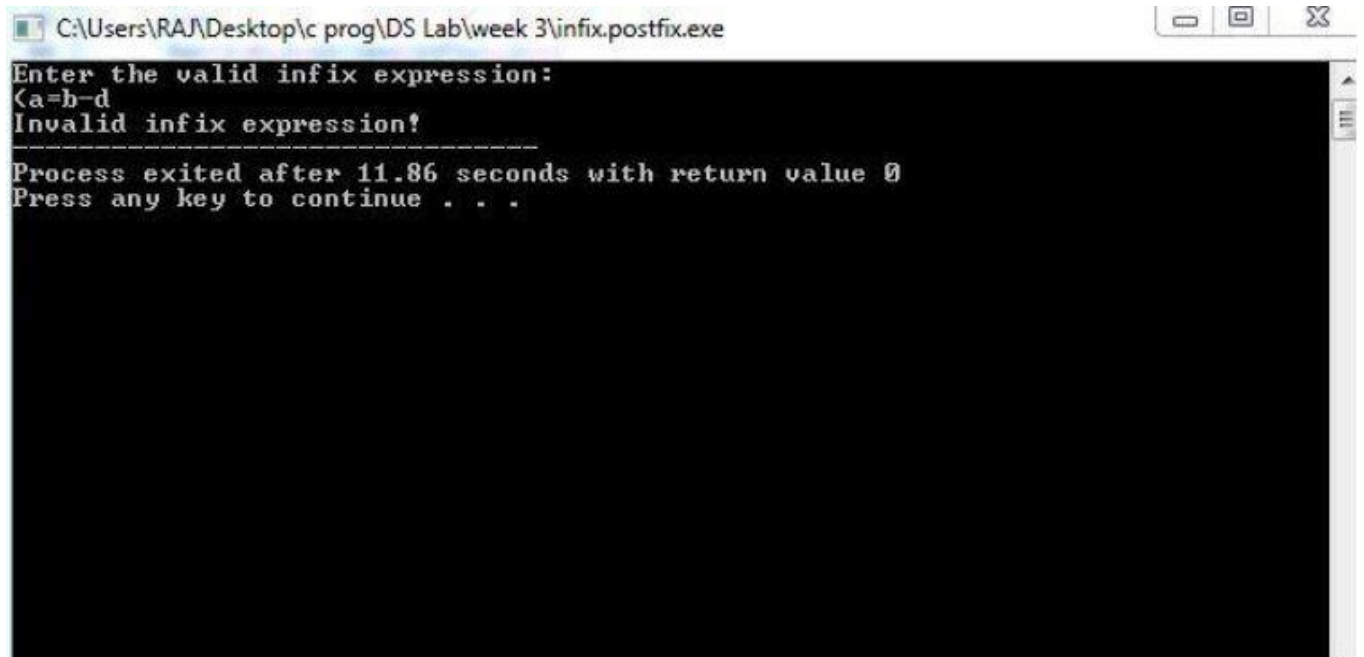
scanf("%s",infix);
for(k=0;k<strlen(infix);k++)
{
    if((isalnum(infix[k])&&isalnum(infix[k+1])))
        flag++;

    if(infix[k]=='(')
        a++;
    if(infix[k]==')')
        b++;
}
if(a!=b||flag!=0)
{
    printf("Invalid infix expression!");
    exit(0);
}
infix_postfix(infix,postfix);
printf("The postfix expression is:\n");
printf("%s\n",postfix);
return 0;
}

```

```
C:\Users\RAJ\Desktop\c prog\DS Lab\week 3\infix.postfix.exe
Enter the valid infix expression:
ab+
Invalid infix expression!
-----
Process exited after 8.2 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\RAJ\Desktop\c prog\DS Lab\week 3\infix.postfix.exe
Enter the valid infix expression:
a^b+c-d+e/f/(g+h)
The postfix expression is:
ab^c+d-ef/gh+/+
-----
Process exited after 86.85 seconds with return value 0
Press any key to continue . . .
```



A screenshot of a Windows command prompt window. The title bar at the top shows the file path: C:\Users\RAJ\Desktop\c prog\DS Lab\week 3\infix.postfix.exe. The window has standard Windows window controls (minimize, maximize, close) on the right. The command prompt text is as follows:
Enter the valid infix expression:
<a=b-d
Invalid infix expression!

Process exited after 11.86 seconds with return value 0
Press any key to continue . . .

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

```
#include<stdio.h>
#include<stdlib.h>
#define QUE_SIZE 5
int item,front=0,rear=-1,q[10];
void insertrear()
{
    if(rear==QUE_SIZE-1)
    {
        printf("\nQueue is full//Overflow\n");
        return;
    }
    printf("\nEnter the item to be inserted\n");
    scanf("%d",&item);
    rear=rear+1;
    q[rear]=item;
}
int deletefront()
{
    if (front>rear)
    {
        front=0;
        rear=-1;
        return -1;
    }
    return q[front++];
}
void displayQ()
{
    int i;
    if (front>rear)
    {
```

```

        printf("\nQueue is empty\n");
        return;
    }
    printf("\nContents of queue\n");
    for(i=front;i<=rear;i++)
    {
        printf("%d ",q[i]);
    }
}
int main()
{
    int choice;
    for(;;)
    {
        printf("\n1:Insert rear\n2:Delete front\n3:Display\n4:exit\n");
        printf("Enter the choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:    insertrear ();
                      break;
            case 2:    item=deletefront();
                      if(item==-1)
                          printf("\nQueue is empty//Underflow\n");
                      else
                          printf("Item deleted=%d\n",item);
                      break;
            case 3:    displayQ();
                      break;
            default:exit (0);
        }
    }
}

```

```
1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
1

Enter the item to be inserted
12

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
1

Enter the item to be inserted
13

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
1

Enter the item to be inserted
14

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
1

Enter the item to be inserted
15

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
1

Enter the item to be inserted
16

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
3

Contents of queue
12 13 14 15 16
1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
2
Item deleted=12

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
1
```

Queue is full//Overflow

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
3

Contents of queue

13 14 15 16
1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
4

Process exited after 26.55 seconds with return value 0
Press any key to continue . . .

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

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

#define que_size 5
int item,front=0,rear=-1,q[que_size],count=0;
void insertrear()
{
    if(count==que_size)
    {
        printf("\nQueue is full//Overflow\n");
        return;
    }
    printf("\nEnter the item to be inserted: ");
    scanf("%d",&item);
    rear=(rear+1)%que_size;
    q[rear]=item;
    count++;
}
int deletefront()
{
    if(count==0) return -1;
    item = q[front];
    front=(front+1)%que_size;
    count=count-1;
    return item;
}
void displayq()
{
    int i,f;
    if(count==0)
```

```

{
printf("\nQueue is empty\n");
return;
}
f=front;
printf("\nContents of queue \n");
for(i=0;i<count;i++)
{
printf("%d\n",q[f]);
f=(f+1)%que_size;
}
}
void main()
{
int choice;
for(;;)
{
printf("\n1.Insert rear \n2.Delete front \n3.Display \n4.Exit \n");
printf("Enter the choice : ");
scanf("%d",&choice);
switch(choice)
{
case 1:insertrear();
break;
case 2:item=deletefront();
if(item==-1)
printf("\nQueue is empty//Underflow\n");
else
printf("Item deleted is: %d \n",item);
break;
case 3:displayq();
break;
default:exit(0);
}
}
}

```

```
1.Insert rear
2.Delete front
3.Display
4.Exit
Enter the choice : 1
```

```
Enter the item to be inserted: 12
```

```
1.Insert rear
2.Delete front
3.Display
4.Exit
Enter the choice : 1
```

```
Enter the item to be inserted: 13
```

```
1.Insert rear
2.Delete front
3.Display
4.Exit
Enter the choice : 1
```

```
Enter the item to be inserted: 14
```

```
1.Insert rear
2.Delete front
3.Display
4.Exit
Enter the choice : 1
```

```
Enter the item to be inserted: 15
```

```
1.Insert rear
2.Delete front
3.Display
4.Exit
Enter the choice : 1
```

```
Enter the item to be inserted: 16
```

```
1.Insert rear
2.Delete front
3.Display
4.Exit
Enter the choice : 1
```

```
Queue is full//Overflow
```

```
1.Insert rear
2.Delete front
3.Display
4.Exit
Enter the choice : 3
```

```
Contents of queue
```

```
12
13
14
15
16
```

```
1.Insert rear
2.Delete front
3.Display
4.Exit
Enter the choice : 2
Item deleted is: 12
```

```
1.Insert rear
2.Delete front
3.Display
4.Exit
Enter the choice : 1
```

Enter the item to be inserted: 17

- 1.Insert rear
- 2.Delete front
- 3.Display
- 4.Exit

Enter the choice : 3

Contents of queue

13
14
15
16
17

- 1.Insert rear
- 2.Delete front
- 3.Display
- 4.Exit

Enter the choice : 4

Process exited after 32.79 seconds with return value 0

Press any key to continue . . .

Lab Program 5&6:

WAP to Implement Singly Linked List with following operations a) 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.

WAP to Implement Singly Linked List with following operations a) 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.

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

typedef struct node *NODE;

NODE getNode()
{
    NODE temp;
    temp = (NODE)malloc(sizeof(struct node));
    if(temp == NULL)
    {
        printf("Memory is full\n");
        return NULL;
    }
    return temp;
}

void freeNode(NODE temp)
{
    free(temp);
}

NODE insertFront(NODE first)
```

```

{
    int num;
    printf("Enter the number\n");
    scanf("%d",&num);
    NODE temp;
    temp = getNode();
    temp->info = num;
    temp->next = NULL;
    if(first==NULL)
    {
        return temp;
    }
    temp->next = first;
    first = temp;
    return first;
}

```

NODE insertRear(NODE first)

```

{
    int num;
    printf("Enter the number\n");
    scanf("%d",&num);
    NODE curr,temp;
    temp = getNode();
    temp->info = num;
    temp->next = NULL;
    if(first == NULL)
    {
        return temp;
    }
    curr = first;
    while(curr->next != NULL)
    {
        curr=curr->next;
    }
    curr->next = temp;
    return first;
}

```

```

}

NODE insertPos(NODE first,int pos)
{
    int num;
    printf("Enter the number\n");
    scanf("%d",&num);
    NODE temp,curr,prev;
    int count=1;
    temp = getNode();
    temp->info = num;
    temp->next = NULL;
    if(pos==count)
    {
        return temp;
    }
    curr = first;
    prev = NULL;
    while(curr!=NULL)
    {
        if(count==pos)
        {
            break;
        }
        prev=curr;
        curr=curr->next;
        count++;
    }
    if(curr==NULL)
    {
        printf("Entered position is more than length\n");
        return first;
    }
    temp->next = prev->next;
    prev->next = temp;
    return first;
}

```

```

void display(NODE first)
{
    if(first == NULL)
    {
        printf("List is empty\n");
        return;
    }
    NODE curr = first;
    while(curr!=NULL)
    {
        printf("%d ",curr->info);
        curr = curr->next;
    }
    printf("\n");
}

NODE deleteFront(NODE first)
{
    NODE curr;
    if(first == NULL)
    {
        printf("List is empty\n");
        return first;
    }
    if(first->next == NULL)
    {
        printf("Deleted element = %d\n",first->info);
        freeNode(first);
        return NULL;
    }
    curr=first;
    curr=curr->next;
    printf("Deleted element = %d\n",first->info);
    freeNode(first);
    return curr;
}

```



```

NODE deleteRear(NODE first)
{
    NODE curr,prev=NULL;
    if(first == NULL)
    {
        printf("List is empty\n");
        return first;
    }
    if(first->next == NULL)
    {
        printf("Deleted element = %d\n",first->info);
        freeNode(first);
        return NULL;
    }
    curr = first;
    while(curr->next!=NULL)
    {
        prev = curr;
        curr=curr->next;
    }
    prev->next = NULL;
    printf("Deleted element = %d\n",curr->info);
    freeNode(curr);
    return first;
}

```

```

NODE deletePos(NODE first,int pos)
{
    NODE curr,prev;
    if(first==NULL)
    {
        printf("List is empty\n");
        return first;
    }
    int count=1;
    if(pos==count)

```

```

    {
        printf("Deleted element = %d\n",first->info);
        curr = first;
        curr = curr->next;
        freeNode(first);
        return curr;
    }
    curr = first;
    prev = NULL;
    while(curr!=NULL)
    {
        if(count==pos)
        {
            break;
        }
        prev = curr;
        curr = curr->next;
        count++;
    }
    if(curr==NULL)
    {
        printf("Entered position is greater than length\n");
        return first;
    }
    prev->next = curr->next;
    printf("Deleted element = %d\n",curr->info);
    freeNode(curr);
    return first;
}

int main()
{
    int ch;
    int pos;
    NODE first = NULL;
    while(1)
    {

```

```

        printf("Enter the option\n1-insertFront\n2-insertRear\n3-
insertPosition\n4-deletedFront\n5-deletedRear\n6-deletePosition\n7-
display\n8-exit\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                first = insertFront(first);
                break;
            case 2:
                first = insertRear(first);
                break;
            case 3:
                printf("Enter the position for insertion\n");
                scanf("%d",&pos);
                first = insertPos(first,pos);
                break;
            case 4:
                first = deleteFront(first);
                break;
            case 5:
                first = deleteRear(first);
                break;
            case 6:
                printf("Enter the position for deletion\n");
                scanf("%d",&pos);
                first = deletePos(first,pos);
                break;
            case 7:
                display(first);
                break;
            case 8:
                return 0;
                break;
        }
    }
}

```

```

Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
List is empty
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
1
Enter the number
20
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
1
Enter the number
30
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
30 20
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
2
Enter the number
10
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
30 20 10
Enter the option
1-insertFront
2-insertRear

```

```
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
1
Enter the number
50
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
50 30 20 10
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
3
Enter the position for insertion
2
Enter the number
40
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
50 40 30 20 10
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
```

```

50 40 30 20 10
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
4
Deleted element = 50
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
5
Deleted element = 10
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
40 30 20
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
6
Enter the position for deletion
2
Deleted element = 30
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
40 20
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
4
Deleted element = 40
Enter the option
1-insertFront
2-insertRear
3-insertPosition

```

```

3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
5
Deleted element = 20
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
4
List is empty
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
5
List is empty
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
7
List is empty
Enter the option
1-insertFront
2-insertRear
3-insertPosition
4-deletedFront
5-deletedRear
6-deletePosition
7-display
8-exit
8
-----
Process exited after 181.2 seconds with return value 0
Press any key to continue . . .

```

Lab Program 7:

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

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

struct node
{
    int info;
    struct node *link;
};
typedef struct node *NODE;

NODE getnode()
{
    NODE x;
    x = (NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
        printf("\nMemory is full\n");
        exit(0);
    }
    return x;
}

NODE insert_front(NODE first,int item)
{
    NODE temp;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(first==NULL)
    {
```



```

        return temp;
    }
    temp->link=first;
    first=temp;
    return first;
}

```

NODE delete_front(NODE first)

```

{
    NODE temp;
    if(first==NULL)
    {
        printf("List is empty. Cannot delete\n");
        return first;
    }
    temp=first;
    temp = temp->link;
    printf("Item deleted at front end is %d\n",first->info);
    free(first);
    return temp;
}

```

NODE IF(NODE second,int item)

```

{
    NODE temp;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(second==NULL)
        return temp;
    temp->link=second;
    second=temp;
    return second;
}

```

NODE IR(NODE second,int item)

```

{

```

```

    NODE temp,cur;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(second==NULL)
        return temp;
    cur=second;
    while(cur->link!=NULL)
        cur=cur->link;
    cur->link=temp;
    return second;
}

```

```

NODE reverse(NODE first)
{
    NODE cur,temp;
    cur=NULL;
    while(first!=NULL)
    {
        temp=first;
        first=first->link;
        temp->link=cur;
        cur=temp;
    }
    return cur;
}

```

```

NODE ascending(NODE first)
{
    NODE prev=first;
    NODE cur=NULL;
    int temp;
    if(first== NULL)
    {
        return 0;
    }
    else

```

```

{
    while(prev!= NULL)
    {
        cur = prev->link;
        while(cur!= NULL)
        {
            if(prev->info > cur->info)
            {
                temp = prev->info;
                prev->info = cur->info;
                cur->info = temp;
            }
            cur = cur->link;
        }
        prev= prev->link;
    }
}
return first;
}

```

NODE descending(NODE first)

```

{
    NODE prev=first;
    NODE cur=NULL;
    int temp;
    if(first==NULL)
    {
        return 0;
    }
    else
    {
        while(prev!= NULL)
        {
            cur = prev->link;
            while(cur!= NULL)
            {
                if(prev->info < cur->info)

```

```

        {
            temp = prev->info;
            prev->info = cur->info;
            cur->info = temp;
        }
        cur = cur->link;
    }
    prev = prev->link;
}
}
return first;
}

```

NODE concatenate(NODE first, NODE second)

```

{
    NODE cur;
    if(first==NULL)
        return second;
    if(second==NULL)
        return first;
    cur=first;
    while(cur->link!=NULL)
    {
        cur=cur->link;
    }
    cur->link=second;
    return first;
}

```

void display(NODE first)

```

{
    NODE temp;
    if(first==NULL)
    {
        printf("List is empty. Cannot display items.\n");
        return;
    }
}

```

```

        printf("List contents are : ");
        for(temp=first;temp!=NULL;temp=temp->link)
        {
            printf("\n%d",temp->info);
        }
    }

void main()
{
    int item,choice,pos,element,option,choice2,item1,num,i;
    NODE first=NULL;
    NODE second=NULL;
    for(;;)
    {
        printf("\n\nChoose an option");
        printf("\n1:Insert_front \n2:Delete_front \n3:Reverse \n4:Sort
\n5.Concatenate \n6:Display \n7:Exit\n");
        printf("Enter the choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("Enter the item at front-end : ");
                    scanf("%d",&item);
                    first=insert_front(first,item);
                    printf("%d inserted at front-end.",first->info);
                    break;
            case 2: first=delete_front(first);
                    break;
            case 3: first=reverse(first);
                    printf("List is reversed.");
                    break;
            case 4: printf("Press 1 for Ascending-sort and 2 for Descending-sort :
");
                    scanf("%d",&option);
                    if(option==1)
                    {
                        first=ascending(first);

```

```

        printf("List is sorted in ascending order.");
    }
    if(option==2)
    {
        first=descending(first);
        printf("List is sorted in descending order.");
    }
    break;
case 5: printf("Create a second list\n");
    printf("Enter the number of elements in the second list : ");
    scanf("%d",&num);
    for(i=1;i<=num;i++)
    {
        printf("\nPress 1 to Insert-front and 2 to Insert-rear : ");
        scanf("%d",&choice2);
        if(choice2==1)
        {
            printf("Enter the item at front-end : ");
            scanf("%d",&item1);
            second=IF(second,item1);
        }
        if(choice2==2)
        {
            printf("Enter the item at rear-end : ");
            scanf("%d",&item1);
            second=IR(second,item1);
        }
    }
    first=concatenate(first,second);
    printf("\nThe two lists are concatenated.");
    break;
case 6: display(first);
    break;
default:exit(0);
    break;
}
}

```

```

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 1
Enter the item at front-end : 0
0 inserted at front-end.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 2
Item deleted at front end is 0

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 2
List is empty. Cannot delete

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 1
Enter the item at front-end : 10
10 inserted at front-end.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 1
Enter the item at front-end : 20
20 inserted at front-end.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 1
Enter the item at front-end : 30
30 inserted at front-end.

```

```

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 6
List contents are :
30
20
10

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 3
List is reversed.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 6
List contents are :
10
20
30

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 4
Press 1 for Ascending-sort and 2 for Descending-sort : 1
List is sorted in ascending order.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 6
List contents are :
10
20
30

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display

```



```

6:Display
7:Exit
Enter the choice : 5
Create a second list
Enter the number of elements in the second list : 0

The two lists are concatenated.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 6
List contents are :
10
20
30

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 4
Press 1 for Ascending-sort and 2 for Descending-sort : 2
List is sorted in descending order.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 6
List contents are :
30
20
10

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 5
Create a second list
Enter the number of elements in the second list : 2

Press 1 to Insert-front and 2 to Insert-rear : 1
Enter the item at front-end : 50

Press 1 to Insert-front and 2 to Insert-rear : 1
Enter the item at front-end : 60

The two lists are concatenated.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort

```

```
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 6
List contents are :
30
20
10
60
50

Choose an option
1:Insert_front
2>Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 7

-----
Process exited after 108.9 seconds with return value 0
Press any key to continue . . .
```

Lab Program 8:

WAP to implement Stack & Queues using Linked Representation

```
#include<stdio.h>
#include<malloc.h>
struct node{
    int val;
    struct node *next;
};
typedef struct node *NODE;
NODE getNode(){
    NODE temp = (NODE)malloc(sizeof(struct node));
    return temp;
}
void freeNode(NODE temp){
    free(temp);
}
NODE push(NODE head){
    NODE temp = getNode();
    printf("Enter the Number\n");
    scanf("%d",&temp->val);
    temp->next = NULL;
    if(head->next == NULL){
        head->next = temp;
        return head;
    }
    NODE curr = head->next;
    while(curr->next != NULL){
        curr = curr->next;
    }
    curr->next = temp;
    return head;
}
NODE pop(NODE head){
    if(head->next == NULL){
```

```

printf("Stack Underflow\n");
return head;
}
NODE curr = head->next,prev=NULL;
if(curr->next==NULL){
printf("Deleted=%d\n",curr->val);
freeNode(curr);
head->next=NULL;
return head;
}
while(curr->next!=NULL){
prev = curr;
curr = curr->next;
}
printf("Deleted=%d\n",curr->val);
prev->next = NULL;
freeNode(curr);
return head;
}
NODE deleteFront(NODE head){
if(head->next==NULL){
printf("Queue Underflow\n");
return head;
}
NODE curr = head->next;
NODE next = curr->next;
head->next = next;
printf("Deleted=%d\n",curr->val);
freeNode(curr);
return head;
}
void display(NODE head,int val){
NODE curr = head->next;
if(val==1){
if(curr==NULL){
printf("Stack is empty\n");
return;
}
}
}

```

```

    }
    printf("Stack Contents:\n");
    while(curr!=NULL){
        printf("%d ",curr->val);
        curr = curr->next;
    }
    printf("\n");
    }else if(val==2){
        if(curr==NULL){
            printf("Queue is empty\n");
            return;
        }
        printf("Queue Contents:\n");
        while(curr!=NULL){
            printf("%d ",curr->val);
            curr = curr->next;
        }
        printf("\n");
    }
}

int main(){
    NODE head = getNode();
    head->next = NULL;
    int chq,ch;
    printf("Enter\n1-Stack\n2-Queue\n");
    scanf("%d",&ch);
    while(1){
        if(ch==1){
            printf("\nEnter the option\n1-push\n2-pop\n3-display\n4-exit\n");
            scanf("%d",&chq);
            switch(chq){
                case 1:
                    head = push(head);
                    break;
                case 2:
                    head = pop(head);
                    break;

```

```

    case 3:
        display(head,ch);
        break;
    case 4:
        return 0;
    }
    }else{
        printf("\nEnter the option\n1-insertRear\n2-deleteFront\n3-display\n4-
        exit\n");
        scanf("%d",&chq);
        switch(chq){
            case 1:
                head = push(head);
                break;
            case 2:
                head = deleteFront(head);
                break;
            case 3:
                display(head,ch);
                break;
            case 4:
                return 0;
        }
    }
    }
    return 0;
}

```

```
Enter
1-Stack
2-Queue
1
Enter the option
1-push
2-pop
3-display
4-exit
2
Stack Underflow
Enter the option
1-push
2-pop
3-display
4-exit
3
Stack is empty
Enter the option
1-push
2-pop
3-display
4-exit
1
Enter the Number
10
Enter the option
1-push
2-pop
3-display
4-exit
1
Enter the Number
20
Enter the option
1-push
2-pop
3-display
4-exit
1
Enter the Number
30
Enter the option
1-push
2-pop
3-display
4-exit
3
Stack Contents:
10 20 30
Enter the option
1-push
2-pop
3-display
4-exit
2
Deleted=30
Enter the option
1-push
2-pop
3-display
4-exit
3
```

Stack Contents:

10 20

Enter the option

1-push

2-pop

3-display

4-exit

4

Process exited after 17.15 seconds with return value 0

Press any key to continue . . .


```

Enter
1-Stack
2-Queue
2

Enter the option
1-insertRear
2-deleteFront
3-display
4-exit
2
Queue Underflow

Enter the option
1-insertRear
2-deleteFront
3-display
4-exit
3
Queue is empty

Enter the option
1-insertRear
2-deleteFront
3-display
4-exit
1
Enter the Number
10

Enter the option
1-insertRear
2-deleteFront
3-display
4-exit
1
Enter the Number
20

Enter the option
1-insertRear
2-deleteFront
3-display
4-exit
1
Enter the Number
30

Enter the option
1-insertRear
2-deleteFront
3-display
4-exit
3
Queue Contents:
10 20 30

Enter the option
1-insertRear
2-deleteFront
3-display
4-exit
2
Deleted=10

Enter the option
1-insertRear
2-deleteFront
3-display
4-exit
3
Queue Contents:
20 30

```

Enter the option

1-insertRear

2-deleteFront

3-display

4-exit

4

Process exited after 20.39 seconds with return value 0

Press any key to continue . . .

Lab Program 9:

Write a program to implement doubly linked list with primitive operations: a) Create a doubly linked list b) Insert nodes at both ends c) Delete nodes at both ends d) Insert a new node to the left of the specified node e) Insert a new node to the right of the specified node f) Delete all key elements g) Display the contents of the list

```
#include<stdio.h>
#include<process.h>
struct node
{
int info;
struct node *llink;
struct node *rlink;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("Memory is full.\n");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
}
NODE dinsert_front(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
```

```

temp->info=item;
cur=head->rlink;
head->rlink=temp;
temp->llink=head;
temp->rlink=cur;
cur->llink=temp;
return head;
}
NODE dinsert_rear(int item,NODE head)
{
    NODE temp,cur;
    temp=getnode();
    temp->info=item;
    cur=head->llink;
    head->llink=temp;
    temp->rlink=head;
    temp->llink=cur;
    cur->rlink=temp;
    return head;
}
NODE ddelete_front(NODE head)
{
    NODE cur,next;
    if(head->rlink==head)
    {
        printf("List is empty.\n");
        return head;
    }
    cur=head->rlink;
    next=cur->rlink;
    head->rlink=next;
    next->llink=head;
    printf("Node deleted is %d",cur->info);
    freenode(cur);
    return head;
}
NODE ddelete_rear(NODE head)

```

```

{
    NODE cur,prev;
    if(head->rlink==head)
    {
        printf("List is empty.\n");
        return head;
    }
    cur=head->llink;
    prev=cur->llink;
    head->llink=prev;
    prev->rlink=head;
    printf("Node deleted is %d",cur->info);
    freenode(cur);
    return head;
}
NODE insert_leftpos(int item,NODE head)
{
    NODE temp,cur,prev;
    if(head->rlink==head)
    {
        printf("List is empty.\n");
        return head;
    }
    cur=head->rlink;
    while(cur!=head)
    {
        if(item==cur->info)
            break;
        cur=cur->rlink;
    }
    if(cur==head)
    {
        printf("Key not found.\n");
        return head;
    }
    prev=cur->llink;
    printf("Enter towards left of %d = ",item);

```

```

temp=getnode();
scanf("%d",&temp->info);
prev->rlink=temp;
temp->llink=prev;
cur->llink=temp;
temp->rlink=cur;
return head;
}
NODE insert_rightpos(int item,NODE head)
{
    NODE temp,cur,prev;
    if(head->rlink==head)
    {
        printf("List is empty.\n");
        return head;
    }
    cur=head->llink;
    while(cur!=head)
    {
        if(item==cur->info)
            break;
        cur=cur->llink;
    }
    if(cur==head)
    {
        printf("Key not found.\n");
        return head;
    }
    prev=cur->rlink;
    printf("Enter towards right of %d = ",item);
    temp=getnode();
    scanf("%d",&temp->info);
    prev->llink=temp;
    temp->rlink=prev;
    cur->rlink=temp;
    temp->llink=cur;
    return head;
}

```

```

}
NODE delete_all_key(int item,NODE head)
{
    NODE prev,cur,next;
    int count;
    if(head->rlink==head)
    {
        printf("List is empty.");
        return head;
    }
    count=0;
    cur=head->rlink;
    while(cur!=head)
    {
        if(item!=cur->info)
            cur=cur->rlink;
        else
        {
            count++;
            prev=cur->llink;
            next=cur->rlink;
            prev->rlink=next;
            next->llink=prev;
            freenode(cur);
            cur=next;
        }
    }
    if(count==0)
        printf("Key not found.");
    else
        printf("Key found at %d positions and are deleted.\n", count);
    return head;
}
void display(NODE head)
{
    NODE temp;
    if(head->rlink==head)

```

```

{
printf("List is empty.\n");
return;
}
printf("Contents of the list : \n");
temp=head->rlink;
while(temp!=head)
{
printf("%d ",temp->info);
temp=temp->rlink;
}
printf("\n");
}
void main()
{
NODE head,last;
int item, choice;
head=getnode();
head->rlink=head;
head->llink=head;
for(;;)
{
printf("\n\n1:Insert front\n2:Insert rear\n3>Delete front\n4>Delete
rear\n5:Insert left position\n6:Insert right position\n7>Delete all key
elements\n8:Display\n9:Exit\n");
printf("Enter the choice : ");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("Enter the item to be inserted at front end : ");
scanf("%d",&item);
last=dinsert_front(item,head);
break;
case 2: printf("Enter the item to be inserted at rear end : ");
scanf("%d",&item);
last=dinsert_rear(item,head);
break;

```



```
case 3: last=ddelete_front(head);
break;
case 4: last=ddelete_rear(head);
break;
case 5: printf("Enter the key item : ");
scanf("%d",&item);
head=insert_leftpos(item,head);
break;
case 6: printf("Enter the key item : ");
scanf("%d",&item);
head=insert_rightpos(item,head);
break;
case 7: printf("Enter the key item : ");
scanf("%d",&item);
head=delete_all_key(item,head);
break;
case 8: display(head);
break;
default:exit(0);
}
}
}
```

```

1:Insert front
2:Insert rear
3>Delete front
4>Delete rear
5:Insert left position
6:Insert right position
7>Delete all key elements
8:Display
9:Exit
Enter the choice : 1
Enter the item to be inserted at front end : 1

1:Insert front
2:Insert rear
3>Delete front
4>Delete rear
5:Insert left position
6:Insert right position
7>Delete all key elements
8:Display
9:Exit
Enter the choice : 2
Enter the item to be inserted at rear end : 13

1:Insert front
2:Insert rear
3>Delete front
4>Delete rear
5:Insert left position
6:Insert right position
7>Delete all key elements
8:Display
9:Exit
Enter the choice : 1
Enter the item to be inserted at front end : 12

1:Insert front
2:Insert rear
3>Delete front
4>Delete rear
5:Insert left position
6:Insert right position
7>Delete all key elements
8:Display
9:Exit
Enter the choice : 8
Contents of the list :
12 1 13

1:Insert front
2:Insert rear
3>Delete front
4>Delete rear
5:Insert left position
6:Insert right position
7>Delete all key elements
8:Display
9:Exit
Enter the choice : 3
Node deleted is 12

1:Insert front
2:Insert rear
3>Delete front
4>Delete rear
5:Insert left position
6:Insert right position

```

```
7:Delete all key elements
8:Display
9:Exit
Enter the choice : 4
Node deleted is 13

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert left position
6:Insert right position
7:Delete all key elements
8:Display
9:Exit
Enter the choice : 1
Enter the item to be inserted at front end : 13

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert left position
6:Insert right position
7:Delete all key elements
8:Display
9:Exit
Enter the choice : 1
Enter the item to be inserted at front end : 14

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert left position
6:Insert right position
7:Delete all key elements
8:Display
9:Exit
Enter the choice : 1
Enter the item to be inserted at front end : 15

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert left position
6:Insert right position
7:Delete all key elements
8:Display
9:Exit
Enter the choice : 2
Enter the item to be inserted at rear end : 12

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert left position
6:Insert right position
7:Delete all key elements
8:Display
9:Exit
Enter the choice : 8
Contents of the list :
15 14 13 1 12

1:Insert front
```

```

2:Insert rear
3>Delete front
4>Delete rear
5:Insert left position
6:Insert right position
7>Delete all key elements
8:Display
9:Exit
Enter the choice : 6
Enter the key item : 12
Enter towards right of 12 = 11

1:Insert front
2:Insert rear
3>Delete front
4>Delete rear
5:Insert left position
6:Insert right position
7>Delete all key elements
8:Display
9:Exit
Enter the choice : 5
Enter the key item : 15
Enter towards left of 15 = 16

1:Insert front
2:Insert rear
3>Delete front
4>Delete rear
5:Insert left position
6:Insert right position
7>Delete all key elements
8:Display
9:Exit
Enter the choice : 7
Enter the key item : 1
Key found at 1 positions and are deleted.

1:Insert front
2:Insert rear
3>Delete front
4>Delete rear
5:Insert left position
6:Insert right position
7>Delete all key elements
8:Display
9:Exit
Enter the choice : 8
Contents of the list :
16 15 14 13 12 11

1:Insert front
2:Insert rear
3>Delete front
4>Delete rear
5:Insert left position
6:Insert right position
7>Delete all key elements
8:Display
9:Exit
Enter the choice : 5
Enter the key item : 5
Key not found.

1:Insert front
2:Insert rear
3>Delete front
4>Delete rear

```

```
3:Delete front
4:Delete rear
5:Insert left position
6:Insert right position
7:Delete all key elements
8:Display
9:Exit
Enter the choice :
6
Enter the key item : 6
Key not found.

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert left position
6:Insert right position
7:Delete all key elements
8:Display
9:Exit
Enter the choice : 7
Enter the key item : 7
Key not found.

1:Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Insert left position
6:Insert right position
7:Delete all key elements
8:Display
9:Exit
Enter the choice : 9

-----
Process exited after 98.81 seconds with return value 0
Press any key to continue . . .
```

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 post order c) To display the elements in the tree

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
#include<process.h>
struct node
{
    int info;
    struct node *rlink;
    struct node *llink;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
        printf("mem full\n");
        exit(0);
    }
    return x;
}
void freenode(NODE x)
{
    free(x);
}
NODE insert(NODE root,int item)
{
    NODE temp,cur,prev;
    temp=getnode();
```

```

temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
if(root==NULL)
    return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
    prev=cur;
    cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
    prev->llink=temp;
else
    prev->rlink=temp;
return root;
}

void display(NODE root,int i)
{
    int j;
    if(root!=NULL)
    {
        display(root->rlink,i+1);
        for(j=0;j<i;j++)
            printf(" ");
        printf("%d\n",root->info);
        display(root->llink,i+1);
    }
}

void preorder(NODE root)
{
    if(root!=NULL)
    {
        printf("%d\n",root->info);
    }
}

```

```

    preorder(root->llink);
    preorder(root->rlink);
}
}
void postorder(NODE root)
{
if(root!=NULL)
{

    postorder(root->llink);
    postorder(root->rlink);
    printf("%d\n",root->info);
}
}
void inorder(NODE root)
{
if(root!=NULL)
{

    inorder(root->llink);
    printf("%d\n",root->info);
    inorder(root->rlink);
}
}
int main()
{
int item,choice;
NODE root=NULL;

for(;;)
{
printf("\n1.insert\n2.display\n3.pre\n4.post\n5.in\n6.exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item\n");

```



```
        scanf("%d",&item);
        root=insert(root,item);
        break;
case 2:display(root,0);
        break;
case 3:preorder(root);
        break;
case 4:postorder(root);
        break;
case 5:inorder(root);
        break;

case 6:exit(0);
        break;
default: continue;
    }
}
```

```
1.insert
2.display
3.pre
4.post
5.in
6.exit
enter the choice
1

enter the item
12

1.insert
2.display
3.pre
4.post
5.in
6.exit
enter the choice
1

enter the item
10

1.insert
2.display
3.pre
4.post
5.in
6.exit
enter the choice
1

enter the item
14

1.insert
2.display
3.pre
4.post
5.in
6.exit
enter the choice
1

enter the item
8

1.insert
2.display
3.pre
4.post
5.in
6.exit
enter the choice
1

enter the item
11

1.insert
2.display
3.pre
4.post
5.in
6.exit
enter the choice
2
```

```

    14
12    11
    10
    8

1.insert
2.display
3.pre
4.post
5.in
6.exit
enter the choice
1

enter the item
13

1.insert
2.display
3.pre
4.post
5.in
6.exit
enter the choice
1

enter the item
20

1.insert
2.display
3.pre
4.post
5.in
6.exit
enter the choice
2

    20
    14
    13
12    11
    10
    8

1.insert
2.display
3.pre
4.post
5.in
6.exit
enter the choice
3

12
10
8
11
14
13
20

1.insert
2.display
3.pre
4.post
5.in
6.exit
enter the choice
4

```

```
8
11
10
13
20
14
12

1.insert
2.display
3.pre
4.post
5.in
6.exit
enter the choice
5

8
10
11
12
13
14
20

1.insert
2.display
3.pre
4.post
5.in
6.exit
enter the choice
6
```