

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4  struct node
5  {
6  int info;
7  struct node*llink;
8  struct node*rlink;
9  };
10 typedef struct node*NODE;
11 NODE getnode()
12 {
13 NODE x;
14 x=(NODE)malloc(sizeof(struct node));
15 if(x==NULL)
16 {
17 printf("Memory not available!");
18 exit(0);
19 }
20 return x;
21 }
22 void freenode(NODE x)
23 {
24 free(x);
25 }
26 NODE insert(int item,NODE root)
27 {
28 NODE temp,cur,prev;
29 char direction[10];
30 int i;
31 temp=getnode();
32 temp->info=item;
33 temp->llink=NULL;
34 temp->rlink=NULL;
35 if(root==NULL)
36 | return temp;
37 printf("Give direction to insert..\n");
38 scanf("%s",direction);
39 prev=NULL;
40 cur=root;
41 for(i=0;i<strlen(direction)&&cur!=NULL;i++)
42 {
43 prev=cur;
44 if(direction[i]=='l')

```

```

44  if(direction[i]=='l')
45  cur=cur->llink;
46  else
47  cur=cur->rlink;
48  }
49  if(cur!=NULL||i!=strlen(direction))
50  {
51  printf("Insertion not possible\n");
52  freenode(temp);
53  return(root);
54  }
55  if(cur==NULL)
56  {
57  if(direction[i-1]=='l')
58  prev->llink=temp;
59  else
60  prev->rlink=temp;
61  }
62  return(root);
63  }
64  void preorder(NODE root)
65  {
66  if(root!=NULL)
67  {
68  printf("the item is %d\n",root->info);
69  preorder(root->llink);
70  preorder(root->rlink);
71  }
72  }
73  void inorder(NODE root)
74  {
75  if(root!=NULL)
76  {
77  inorder(root->llink);
78  printf("The item is%d\n",root->info);
79  inorder(root->rlink);
80  }
81  }
82  void postorder(NODE root)
83  {
84  if (root!=NULL)
85  {
86  postorder(root->llink);

```

```

86 postorder(root->llink);
87 postorder(root->rlink);
88 printf("The item is%d\n",root->info);
89 }
90 }
91 void display(NODE root,int i)
92 {
93     int j;
94     if(root!=NULL)
95     {
96         display(root->rlink,i+1);
97         for (j=1;j<=i;j++)
98             printf(" ");
99         printf("%d\n",root->info);
100        display(root->llink,i+1);
101    }
102 }
103
104 int main()
105 {
106     NODE root=NULL;
107     int choice,i,item;
108
109     for(;;)
110     {
111         printf("1.Insert\n2.Preorder\n3.Inorder\n4.Postorder\n5.Display\n");
112         printf("Enter the choice:\n");
113         scanf("%d",&choice);
114         switch(choice)
115         {
116             case 1: printf("Enter the item:\n");
117                     scanf("%d",&item);
118                     root=insert(item,root);
119                     break;
120             case 2: if(root==NULL)
121                     {
122                         printf("Tree is empty!");
123                     }
124                     else
125                     {
126                         printf("Given tree is..");
127                         display(root,1);
128                         printf("The preorder traversal is:\n");

```

```

{
    printf("Given tree is..");
    display(root,1);
    printf("The preorder traversal is:\n");
    preorder(root);
}
break;
case 3:if(root==NULL)
{
    printf("Tree is empty");
}
else
{
    printf("Given tree is..");
    display(root,1);
    printf("The inorder traversal is \n");
    inorder(root);
}
break;
case 4:if (root==NULL)
{
    printf("Tree is empty");
}
else
{
    printf("Given tree is..");
    display(root,1);
    printf("The postorder traversal is \n");
    postorder(root);
}
break;
case 5:display(root,1);
break;
default:printf("Invalid choice entered.\n");
        exit(0);
}
}
return 0;
}

```


➤ clang-7 -pthread -lm -o main main.c

➤ ./main

1.Insert

2.Preorder

3.Inorder

4.Postorder

5.Display

Enter the choice:

1

Enter the item:

21

1.Insert

2.Preorder

3.Inorder

4.Postorder

5.Display

Enter the choice:

1

Enter the item:

33

Give direction to insert..

l

1.Insert

2.Preorder

3.Inorder

4.Postorder

5.Display

Enter the choice:

1

Enter the item:

54

Give direction to insert..

r

1.Insert

2.Preorder

3.Inorder

4.Postorder

5.Display

Enter the choice:

```
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
77
Give direction to insert..
ll
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
56
Give direction to insert..
lr
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
92
Give direction to insert..
rl
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
83
Give direction to insert..
rr
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
```

1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:

5
83
54
92
21
56
33
77

1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:

2
Given tree is.. 83
54
92
21
56
33
77

The preorder traversal is:

the item is 21
the item is 33
the item is 77
the item is 56
the item is 54
the item is 92
the item is 83

1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:

4.Postorder
 5.Display
 Enter the choice:
 3
 Given tree is.. 83
 54
 92
 21
 56
 33
 77

The inorder traversal is
 The item is77
 The item is33
 The item is56
 The item is21
 The item is92
 The item is54
 The item is83

1.Insert
 2.Preorder
 3.Inorder
 4.Postorder
 5.Display
 Enter the choice:
 4

Given tree is.. 83
 54
 92
 21
 56
 33
 77

The postorder traversal is
 The item is77
 The item is56
 The item is33
 The item is92
 The item is83
 The item is54
 The item is21

1.Insert
 2.Preorder
 3.Inorder
 4.Postorder
 5.Display
 Enter the choice:

LAB PROGRAM - 10

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

struct node
{
    int info;
    struct node *link;
    struct node *slink;
};

typedef struct node * NODE;
NODE getnode()
{
    NODE x;
    x = (NODE) malloc (size of (struct node));
    if (x == NULL)
    {
        printf("Memory not available!");
        exit(0);
    }
    return x;
}

void free_node(NODE x)
{
    free(x);
}

NODE insert(int item, NODE root)
{
    NODE temp, cur, prev;
    char direction[10];
    int i;
    temp = getnode();
    temp->info = item;
    temp->link = NULL;
    temp->slink = NULL;
    if (root == NULL)
    {
        return temp;
    }
    printf("Give direction to insert: \n");
    scanf("%s", direction);
    prev = NULL;
    cur = root;
    for (i = 0; i < strlen(direction) && cur != NULL; i++)
```

```
{ prev = cur;
if (direction[i] == 'I')
cur = cur -> link;
else
cur = cur -> rlink; }
if (cur != NULL || i != strlen(direction))
{
```

```
printf("Insertion not possible\n");
freemem(temp);
return root;
}
```

```
if (cur == NULL)
{ if (direction[i-1] == 'I')
prev -> link = temp;
else
```

```
prev -> rlink = temp; }
return root; }
```

```
void preorder (NODE root)
```

```
{ if (root != NULL)
```

```
{ printf("The item is %d\n", root -> info);
```

```
preorder (root -> link);
```

```
preorder (root -> rlink); }
```

```
void inorder (NODE root)
```

```
{
```

```
if (root != NULL)
```

```
{ inorder (root -> link);
```

```
printf("The item is %d\n", root -> info);
```

```
inorder (root -> rlink); }
```

```
void postorder (NODE root)
```

```
{
```

```
if (root != NULL)
```

```
{ postorder (root -> link);
```

```
postorder (root -> rlink);
```

```
printf("The item is %d\n", root -> info);
```

```
}
```

```
}
```



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

int main()
{
    NODE root = NULL;
    int choice, i, item;
    for (;;)
    {
        printf("1. Insert\n 2. Preorder\n 3. Inorder\n 4. Postorder\n 5. Display\n");
        printf("Enter the choice:\n");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1: printf("Enter the item:\n");
                    scanf("%d", &item);
                    root = insert (item, root);
                    break;
            case 2: if (root == NULL)
                    { printf("Tree is empty!"); }
                    else {
                        printf("Given tree is:\n");
                        display (root, 1);
                        printf("The preorder traversal is:\n");
                        preorder (root);
                    }
                    break;
            case 3: if (root == NULL)
                    { printf("Tree is empty!"); }
                    else {
                        printf("Given tree is:\n");
                        display (root, 1);
                        printf("The inorder traversal is:\n");
                        inorder (root);
                    }
                    break;
            case 4: if (root == NULL)
                    { printf("Tree is empty!"); }
                    else {
                        printf("Given tree is:\n");
                        display (root, 1);
                        printf("The postorder traversal is:\n");
                        postorder (root);
                    }
                    break;
            case 5: display (root, 1);
                    break;
        }
    }
}
```

S.R. POOJA
1BM19CS135

Date _____
Page _____

```
printf("Given tree is...");  
display(root, 1);  
printf("The inorder traversal is\n");  
inorder(root);  
break;  
case 4: if (root == NULL)  
{ printf("Tree is empty");  
}  
else  
{  
printf("Given tree is...");  
display(root, 1);  
printf("The postorder traversal is\n");  
postorder(root);  
} break;  
case 5: display(root, 1);  
break;  
default: printf("Invalid choice entered.\n");  
getchar();  
}  
}  
return 0;  
}
```