

main.c

```
1  /*BINARY TREE*/
2
3  #include<stdio.h>
4  #include<stdlib.h>
5  #include<string.h>
6  struct node
7  {
8      int info;
9      struct node*llink;
10     struct node*rlink;
11 };
12 typedef struct node*NODE;
13 NODE getnode()
14 {
15     NODE x;
16     x=(NODE)malloc(sizeof(struct node));
17     if(x==NULL)
18     {
19         printf("memory not available");
20         exit(0);
21     }
22     return x;
23 }
24 void freenode(NODE x)
25 {
26     free(x);
27 }
28 NODE insert(int item,NODE root)
29 {
30     NODE temp,cur,prev;
31     char direction[10];
32     int i;
33     temp=getnode();
34     temp->info=item;
35     temp->llink=NULL;
36     temp->rlink=NULL;
37     if(root==NULL)
38         return temp;
39     printf("give direction to insert\n");
```

```
37 if(root==NULL)
38     return temp;
39 printf("give direction to insert\n");
40 scanf("%s",direction);
41 prev=NULL;
42 cur=root;
43 for(i=0;i<strlen(direction)&&cur!=NULL;i++)
44 {
45     prev=cur;
46     if(direction[i]=='l')
47         cur=cur->llink;
48     else
49         cur=cur->rlink;
50 }
51 if(cur!=NULL||i!=strlen(direction))
52 {
53     printf("insertion not possible\n");
54     freenode(temp);
55     return(root);
56 }
57 if(cur==NULL)
58 {
59     if(direction[i-1]=='l')
60         prev->llink=temp;
61     else
62         prev->rlink=temp;
63 }
64 return(root);
65 }
66 void preorder(NODE root)
67 {
68     if(root!=NULL)
69     {
70         printf("%d\n",root->info);
71         preorder(root->llink);
72         preorder(root->rlink);
73     }
74 }
75 void inorder(NODE root)
```

main.c

```
73 }
74 }
75 void inorder(NODE root)
76 {
77     if(root!=NULL)
78     {
79         inorder(root->llink);
80         printf("%d\n",root->info);
81         inorder(root->rlink);
82     }
83 }
84 void postorder(NODE root)
85 {
86     if (root!=NULL)
87     {
88         postorder(root->llink);
89         postorder(root->rlink);
90         printf("%d\n",root->info);
91     }
92 }
93 void display(NODE root,int i)
94 {
95     int j;
96     if(root!=NULL)
97     {
98         display(root->rlink,i+1);
99         for (j=1;j<=i;j++)
100             printf(" ");
101         printf("%d\n",root->info);
102         display(root->llink,i+1);
103     }
104 }
105 void main()
106 {
107     NODE root=NULL;
108     int choice,item;
109     for(;;)
110     {
111         printf("1 insert\n2 preorder\n3 inorder\n4 postorder\n5 display\n");
```

```
108     int choice,item;  
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         printf("-----\n");  
115         switch(choice)  
116         {  
117             case 1: printf("enter the item\n");  
118                     scanf("%d",&item);  
119                     root=insert(item,root);  
120                     break;  
121             case 2: if(root==NULL)  
122                     {  
123                         printf("tree is empty");  
124                     }  
125                     else  
126                     {  
127                         printf("the preorder traversal is \n");  
128                         preorder(root);  
129                     }  
130                     break;  
131             case 3:if(root==NULL)  
132                     {  
133                         printf("tree is empty");  
134                     }  
135                     else  
136                     {  
137                         printf("the inorder traversal is \n");  
138                         inorder(root);  
139                     }  
140                     break;  
141             case 4:if (root==NULL)  
142                     {  
143                         printf("tree is empty");  
144                     }  
145                     else  
146                     {  
147                         printf("the postorder traversal is \n");
```

main.c

```
123     printf("tree is empty");
124 }
125 else
126 {
127     printf("the preorder traversal is \n");
128     preorder(root);
129 }
130 break;
131 case 3: if (root == NULL)
132 {
133     printf("tree is empty");
134 }
135 else
136 {
137     printf("the inorder traversal is \n");
138     inorder(root);
139 }
140 break;
141 case 4: if (root == NULL)
142 {
143     printf("tree is empty");
144 }
145 else
146 {
147     printf("the postorder traversal is \n");
148     postorder(root);
149 }
150 break;
151 case 5: display(root, 1);
152 break;
153 default: exit(0);
154 }
155 }
156 }
```

```
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
-----
enter the item
10
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
-----
enter the item
20
give direction to insert
1
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
-----
enter the item
30
give direction to insert
r
1.insert
2.preorder
3.inorder
```

input

```
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
-----
enter the item
40
give direction to insert
ll
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
-----
enter the item
60
give direction to insert
rl
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
-----
enter the item
50
give direction to insert
lr
1.insert
2.preorder
```

```
enter the choice
1
-----
enter the item
70
give direction to insert
rr
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
-----
      70
     30
    60
   10
  50
 20
 40
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
2
-----
the preorder traversal is
10
20
40
50
30
60
```


input

```
the preorder traversal is
10
20
40
50
30
60
70
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
3
-----
the inorder traversal is
40
20
50
10
60
30
70
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
4
-----
the postorder traversal is
40
50
20
60
```

input

```
-----  
the inorder traversal is
```

```
40  
20  
50  
10  
60  
30  
70
```

```
1.insert  
2.preorder  
3.inorder  
4.postorder  
5.display  
enter the choice  
4
```

```
-----  
the postorder traversal is
```

```
40  
50  
20  
60  
70  
30  
10
```

```
1.insert  
2.preorder  
3.inorder  
4.postorder  
5.display  
enter the choice  
6
```

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

```
1 #include<stdio.h>
2 #include<conio.h>
3 #include<stdlib.h>
4 struct node
5 {
6     int info;
7     struct node *rlink;
8     struct node *llink;
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("mem full\n");
18         exit(0);
19     }
20     return x;
21 }
22 void freenode(NODE x)
23 {
24     free(x);
25 }
26 NODE insert(NODE root,int item)
27 {
28     NODE temp,cur,prev;
29     temp=getnode();
30     temp->rlink=NULL;
31     temp->llink=NULL;
32     temp->info=item;
33     if(root==NULL)
34         return temp;
35     prev=NULL;
36     cur=root;
37     while(cur!=NULL)
38     {
39         prev=cur;
```

main.c

```
37 while(cur!=NULL)
38 {
39     prev=cur;
40     cur=(item<cur->info)?cur->llink:cur->rlink;
41 }
42 if(item<prev->info)
43     prev->llink=temp;
44 else
45     prev->rlink=temp;
46 return root;
47 }
48 void display(NODE root,int i)
49 {
50     int j;
51     if(root!=NULL)
52     {
53         display(root->rlink,i+1);
54         for(j=0;j<i;j++)
55             printf(" ");
56         printf("%d\n",root->info);
57         display(root->llink,i+1);
58     }
59 }
60 void preorder(NODE root)
61 {
62     if(root!=NULL)
63     {
64         printf("%d\n",root->info);
65         preorder(root->llink);
66         preorder(root->rlink);
67     }
68 }
69 void postorder(NODE root)
70 {
71     if(root!=NULL)
72     {
73         postorder(root->llink);
74         postorder(root->rlink);
75         printf("%d\n",root->info);
```

```
76 }
77 }
78 void inorder(NODE root)
79 {
80     if(root!=NULL)
81     {
82         inorder(root->llink);
83         printf("%d\n",root->info);
84         inorder(root->rlink);
85     }
86 }
87 void maximum(NODE root)
88 {
89     while(root!=NULL && root->rlink!=NULL)
90     {
91         root=root->rlink;
92     }
93     printf("\nMaximum value is %d",root->info);
94 }
95 void minimum(NODE root)
96 {
97     while(root!=NULL && root->llink!=NULL)
98     {
99         root=root->llink;
100     }
101     printf("\nMinimum value is %d",root->info);
102 }
103 int count(NODE root)
104 {
105     int c=1;
106     if(root==NULL)
107         return 0;
108     else{
109         c+=count(root->llink);
110         c+=count(root->rlink);
111         return c;
112     }
113 }
114 void main()
```

```
112     }
113 }
114 void main()
115 {
116     int item,choice,c;
117     NODE root=NULL;
118     for(;;)
119     {
120         printf("\n1.insert\n2.display\n3.pre-order\n4.post-order\n5.in-order\n6.count\n7.Minimum element\n8.Maximum element\n9.exit\n");
121         printf("enter the choice\n");
122         scanf("%d",&choice);
123         printf("-----\n");
124         switch(choice)
125         {
126             case 1:printf("enter the item\n");
127                     scanf("%d",&item);
128                     root=insert(root,item);
129                     break;
130             case 2:display(root,0);
131                     break;
132             case 3:preorder(root);
133                     break;
134             case 4:postorder(root);
135                     break;
136             case 5:inorder(root);
137                     break;
138             case 6:c=count(root);
139                     printf("No. of nodes are: %d\n",c);
140                     break;
141             case 7:minimum(root);
142                     break;
143             case 8:maximum(root);
144                     break;
145             default:exit(0);
146                     break;
147         }
148     }
149 }
150 }
```

```
1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
1
-----
enter the item
50

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
1
-----
enter the item
70

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
```

```
enter the choice
1
-----
enter the item
60

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
1
-----
enter the item
90

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
1
-----
enter the item
20

1.insert
```



```
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
1
-----
enter the item
10

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
1
-----
enter the item
40

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
2
```

```
1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
2
```

```
-----
          90
         70
        60
50
       40
      20
     10
```

```
1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
6
```

```
-----
No. of nodes are: 7
```

```
1.insert
2.display
```

```
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
6
-----
```

No. of nodes are: 7

```
1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
7
-----
```

Minimum value is 10

```
1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
8
-----
```