

```
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;
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 NODE delete(NODE root,int item)
61 {
62     NODE cur,parent,q,suc;
63     if(root==NULL)
64     {
65         printf("empty\n");
66         return root;
67     }
68     parent=NULL;
69     cur=root;
70     while(cur!=NULL&&item!=cur->info)
71     {
72         parent=cur;
73         cur=(item<cur->info)?cur->llink:cur->rlink;
74     }
75     if(cur==NULL)
76     {
```





```
77     printf("not found\n");
78     return root;
79 }
80 if(cur->llink==NULL)
81     q=cur->rlink;
82 else if(cur->rlink==NULL)
83     q=cur->llink;
84 else
85 {
86     suc=cur->rlink;
87     while(suc->llink!=NULL)
88         suc=suc->llink;
89     suc->llink=cur->llink;
90     q=cur->rlink;
91 }
92 if(parent==NULL)
93     return q;
94 if(cur==parent->llink)
95     parent->llink=q;
96 else
97     parent->rlink=q;
98 freenode(cur);
99 return root;
100 }
101
102 void preorder(NODE root)
103 {
104     if(root!=NULL)
105     {
106         printf("%d\n",root->info);
107         preorder(root->llink);
108         preorder(root->rlink);
109     }
110 }
111 void postorder(NODE root)
112 {
113     if(root!=NULL)
114     {
```



nodes.c

mm.c

```
116     postorder(root->llink);
117     postorder(root->rlink);
118     printf("%d\n",root->info);
119 }
120 }
121 void inorder(NODE root)
122 {
123     if(root!=NULL)
124     {
125
126         inorder(root->llink);
127         printf("%d\n",root->info);
128         inorder(root->rlink);
129     }
130 }
131 void largest(NODE root)
132 {
133     while (root != NULL && root->rlink != NULL)
134     {
135         root = root->rlink;
136     }
137     printf("\nLargest value is %d", root->info);
138 }
139 void smallest(NODE root)
140 {
141     while (root != NULL && root->llink != NULL)
142     {
143         root = root->llink;
144     }
145     printf("\nSmallest value is %d", root->info);
146 }
147 void main()
148 {
149     int item,choice;
150     NODE root=NULL;
151     for(;;)
152     {
153         printf("\n1.Insert\n2.Display\n3.Pre-order\n4.Post-order\n5.In-order\n6.Delet
```





```
147 void main()
148 {
149     int item,choice;
150     NODE root=NULL;
151     for(;;)
152     {
153         printf("\n1.Insert\n2.Display\n3.Pre-order\n4.Post-order\n5.In-c
154         printf("Enter the choice\n");
155         scanf("%d",&choice);
156         switch(choice)
157         {
158             case 1:printf("Enter the item\n");
159                     scanf("%d",&item);
160                     root=insert(root,item);
161                     break;
162             case 2:printf("Contents of tree:\n");
163                     display(root,0);
164                     break;
165             case 3:preorder(root);
166                     break;
167             case 4:postorder(root);
168                     break;
169             case 5:inorder(root);
170                     break;
171             case 6:printf("Enter the item\n");
172                     scanf("%d",&item);
173                     root=delete(root,item);
174                     break;
175             case 7:largest(root);
176                     break;
177             case 8:smallest(root);
178                     break;
179             default:exit(0);
180                     break;
181         }
182     }
183 }
```