

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

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

```
struct node
```

```
{
```

```
    int info;
```

```
    struct node *rlink;
```

```
    struct node *llink;
```

```
};
```

```
typedef struct node *NODE;
```

```
NODE getnode()
```

```
{
```

```
    NODE x;
```

```
    x = (NODE) malloc(sizeof(NODE));
```

```
    if (x == NULL)
```

```
{
```

```
        printf("memory 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->link : cur->rlink;
    }
    if (item < prev->info)
        prev->link = info; 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->link, i+1);
    }
}

```

```

NODE delete (NODE root, int item)
{
    NODE cur, prev parent, q, r;
    if (root == NULL)
    {

```

```
printf("Empty \n");  
return root;
```

```
}
```

```
parent = NULL;
```

```
cur = root;
```

```
while (cur != NULL && item != cur->info)
```

```
{
```

```
parent = cur;
```

```
cur = (item < cur->info) ? cur->llink : cur->rlink;
```

```
}
```

```
if (cur == NULL)
```

```
{
```

```
printf("Not found \n");
```

```
return root;
```

```
}
```

```
if (cur->llink == NULL)
```

```
q = cur->rlink;
```

```
else if (cur->rlink == NULL)
```

```
q = cur->llink;
```

```
else
```

```
{
```

```
suc = cur->rlink
```

```
while (suc->llink != NULL)
```

```
suc = suc->llink;
```

```
suc->llink = cur->llink;
```

```
q = cur->lrlink;
```

```
}
```

```
freemove (cur);
```

```
return root;
```

```
}
```

Page _____
SPLASH

```
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 root->llink);
```

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

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

```
}
```

```
}
```

```
int main()
```

```
{
```

```
int item, choice;
```

```
NODE root = NULL;
```

```
for(;;)
```


2

```
printf("\n 1. Insert \n 2. Display \n 3. Preorder \n 4. Postorder  
 5. Inorder \n 6. Delete \n 7. Exit \n");
```

```
printf("Enter 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 display(root, 0);
```

```
break;
```

```
case 3: preorder(root);
```

```
break;
```

```
case 4: postorder(root);
```

```
break;
```

```
case 5: inorder(root);
```

```
break;
```

```
case 6: printf("Enter the item \n");
```

```
scanf("%d", &item);
```

```
root = delete(root, item);
```

```
break;
```

```
default: exit(0);
```

```
break;
```

```
}
```

```
}
```

```
}
```

```

1 #include<stdio.h>
2 #include<stdio.h>
3 #include<stdlib.h>
4 struct node {
5     int info;
6     struct node* rlink;
7     struct node* llink;
8 };
9 typedef struct node*NODE;
10 NODE getnode() {
11     NODE x;
12     x=(NODE)malloc(sizeof(struct node));
13     if(x==NULL) {
14         printf("memfull\n");
15         exit(0);
16     }
17     return x;
18 }
19 void freenode(NODE x) {
20     free(x);
21 }
22 NODE insert(NODE root,int item) {
23     NODE temp,cur,prev;
24     temp=getnode();
25     temp->rlink=NULL;
26     temp->llink=NULL;
27     temp->info=item;
28     if(root==NULL)
29         return temp;
30     prev=NULL;
31     cur=root;
32     while(cur!=NULL) {
33         prev=cur;
34         cur=(item<cur->info)?
35         cur->llink:cur->rlink;
36     }
37     if(item<prev->info)
38         prev->llink=temp;
39     else
40         prev->rlink=temp;
41     return root;
42 }
43 void display(NODE root,int i) {
44     int j;
45     if(root!=NULL) {
46         display(root->rlink,i+1);
47         for(j=0;j<i;j++)
48             printf(" ");
49         printf("%d\n",root->info);
50         display(root->llink,i+1);
51     }
52 }
53 NODE deleter(NODE root,int item) {

```

```

53  NODE cur,parent,q,suc;
54  if(root==NULL) {
55      printf("empty\n");
56      return root;
57  }
58  parent=NULL;
59  cur=root;
60  while(cur!=NULL&&item!=cur->info) {
61      parent=cur;
62      cur=(item<cur->info)?
cur->llink:cur->rlink;
63  }
64  if(cur==NULL) {
65      printf("not found\n");
66      return root;
67  }
68  if(cur->llink==NULL)
69      q=cur->rlink;
70  else if(cur->rlink==NULL)
71      q=cur->llink;
72  else {
73      suc=cur->rlink;
74      while(suc->llink!=NULL)
75          suc=suc->llink;
76      suc->llink=cur->llink;
77      q=cur->rlink;
78  }
79  if(parent==NULL)
80      return q;
81  if(cur==parent->llink)
82      parent->llink=q;
83  else
84      parent->rlink=q;
85      freenode(cur);
86      return root;
87  }
88  void preorder(NODE root) {
89      if(root!=NULL) {
90          printf("%d\n",root->info);
91          preorder(root->llink);
92          preorder(root->rlink);
93      }
94  }
95  void postorder(NODE root) {
96      if(root!=NULL) {
97          postorder(root->llink);
98          postorder(root->rlink);
99          printf("%d\n",root->info);
100      }
101  }
102  void inorder(NODE root) {
103      if(root!=NULL) {
104          inorder(root->llink);

```

```

105     printf("%d\n", root->info);
106     inorder(root->rlink);
107 }
108 }
109 int main() {
110     int item, choice;
111     NODE root=NULL;
112     for(;;) {
113
114         printf("\n1.Insert\n2.Display\n3.Pre\n4.Post\n5.In\n6.Delete\n7.Exit\n");
115         printf("Enter the choice\n");
116         scanf("%d",&choice);
117         switch(choice) {
118             case 1:
119                 printf("Enter the item\n");
120                 scanf("%d",&item);
121                 root=insert(root,item);
122                 break;
123             case 2:
124                 display(root,0);
125                 break;
126             case 3:
127                 preorder(root);
128                 break;
129             case 4:
130                 postorder(root);
131                 break;
132             case 5:
133                 inorder(root);
134                 break;
135             case 6:
136                 printf("Enter the item\n");
137                 scanf("%d",&item);
138                 root=deleter(root,item);
139                 break;
140             default:
141                 exit(0);
142         }
143     }

```



```
1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
```

```
1
```

```
Enter the item
10
```

```
1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
```

```
1
```

```
Enter the item
20
```

```
1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
```

```
1
```

```
Enter the item
30
```

```
1.Insert
2.Display
```

3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice

1

Enter the item

9

1.Insert
2.Display
3.Pre

4.Post

5.In

6.Delete

7.Exit

Enter the choice

1

Enter the item

8

1.Insert
2.Display
3.Pre

4.Post

5.In

6.Delete

7.Exit

Enter the choice

1

Enter the item

7

1.Insert
2.Display
3.Pre

4.Post

5.In

6.Delete
7.Exit
Enter the choice
6
Enter the item
7

1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
2
 30
 20
10
 9
 8

1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
3
10
9
8
20
30

1.Insert
2.Display

3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice

4
8
9
30
20
10

1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice

5
8
9
10
20
30