```
#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)
{
pf("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->llink : cur->rlink;
 }
 if (item < prev->info)
  prev->llink = 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->llink, i+1);
 }
 }
NODE delete (NODE root, int item)
{
NODE cur, parent, q, suc;
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->rlink;
}
freenode (cur);
return root;
}
```

```
void preorder(NODE root)
{
 if(root != NULL)
 {
  printf("%d \n", root->info);
  preorder(root->llink);
  preorder(root->llink);
  }
  }
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. Preorder\n4. Postorder
    \n5. Inorder\n6. Delete\n7. 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(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;
  }
 }
}
```

```c
#include<stdio.h>
#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(struct node));
  if(x==NULL) {
    printf("memfull\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);
  }
}
NODE deleter(NODE root,int item) {
```

```
 NODE cur,parent,q,suc;
 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->rlink;
  }
 if(parent==NULL)
  return q;
 if(cur==parent->llink)
  parent->llink=q;
 else
  parent->rlink=q;
  freenode(cur);
  return root;
}
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);
```

```
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
    printf("\n1.Insert\n2.Display\n3.Pre\n4.Post\
    n5.In\n6.Delete\n7.Exit\n");
114     printf("Enter the choice\n");
115     scanf("%d",&choice);
116     switch(choice) {
117       case 1:
118         printf("Enter the item\n");
119         scanf("%d",&item);
120         root=insert(root,item);
121         break;
122       case 2:
123         display(root,0);
124         break;
125       case 3:
126         preorder(root);
127         break;
128       case 4:
129         postorder(root);
130         break;
131       case 5:
132         inorder(root);
133         break;
134       case 6:
135         printf("Enter the item\n");
136         scanf("%d",&item);
137         root=deleter(root,item);
138         break;
139       default:
140         exit(0);
141       }
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
```