

→ DS Lab Program 10 →

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

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

```
struct Node {
```

```
    int info;
```

```
    struct Node *rlink;
```

```
    struct Node *llink;
```

```
};
```

```
typedef struct Node node;
```

```
node getnode();
```

```
node *node();
```

```
n = (node) malloc (sizeof (node));
```

```
if (n == NULL)
```

```
    printf ("Memory full\n");
```

```
exit (0);
```

```
}
```

```
return n;
```

```
}
```

```
void freenode (node n);
```

```
free (n);
```

```
}
```

node insert (node root, int item) {

 node temp, prev, cur;

 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 :

 }

 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 delete (node root, int item) {

 node cur, parent, q, succ;

 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 {
    cur = cur->rlink;
    while (cur->llink != NULL)
        cur = cur->llink;
    cur->llink = cur->rlink;
    q = cur->rlink;
}

if (parent == NULL)
    return q;
if (parent->cur == parent->llink)
    parent->llink = q;
else
    parent->rlink = q;
freemode(cur);
return root;
}

```

```

void preorder(node root) {
    if (root != NULL) {
        printf ("%d\n", root->info);
        preorder (root->llink);
        preorder (root->rlink);
        printf ("%d\n", root->info);
    }
}

```

```

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.\n"
               "Pre \n4. Post \n5. In \n6. Delete \n"
               "7. Exit \n");
        printf ("Enter the choice\n");
        scanf ("%d", &choice);
        switch (choice) {

```

case 1: printf ("Enter 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;

3.

```
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)?cur->llink:cur->rlink;
35     }
36     if(item<prev->info)
37         prev->llink=temp;
38     else
39         prev->rlink=temp;
40     return root;
41 }
42 void display(NODE root,int i) {
43     int j;
44     if(root!=NULL) {
45         display(root->rlink,i+1);
```

```
43     int j;
44     if(root!=NULL) {
45         display(root->rlink,i+1);
46         for(j=0;j<i;j++)
47             printf(" ");
48         printf("%d\n",root->info);
49         display(root->llink,i+1);
50     }
51 }
52 NODE delete(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 }
```

```
85         freenode(cur);
86     }
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         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);
```

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