

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<malloc.h>
4
5  typedef struct treeNode
6  {
7      int data;
8      struct node*left;
9      struct node*right;
10     struct node*key;
11 }Node;
12
13 //Function to create a newnode
14 Node*createnode(int value)
15 {
16     Node*newnode=(Node*)malloc(sizeof(Node));
17     newnode->data=value;
18     newnode->left=NULL;
19     newnode->right=NULL;
20     return newnode;
21 }
22
23 //Function to insert node in binary tree
24 Node*insert(Node*root,int value)
25 {
26     if(root==NULL)
27     {
28         return createnode(value);
29     }
30     if(value<root->data)
31     {
32         root->left=insert(root->left,value);
33     }
34     else if(value>root->data)
35     {
36         root->right=insert(root->right,value);
37     }
38     return root;
39 }
40
41 //In-order traversal (LOR)
42 void inorder(Node *root)
43 {
44     if(root!=NULL)
45     {
46         inorder(root->left);
47         printf("%d\t",root->data);
48         inorder(root->right);
49     }
50 }
51
52 //Pre-order traversal (OLR)
53 void preorder(Node*root)
54 {
55     if(root!=NULL)
56     {
57         printf("%d\t",root->data);
58         preorder(root->left);
59         preorder(root->right);
60     }
61 }
62
63 //Post-order traversal (LRO)
64 void postorder(Node*root)
65 {
66     if(root!=NULL)

```

```

67     {
68         postorder(root->left);
69         postorder(root->right);
70         printf("%d\t",root->data);
71     }
72 }
73
74
75 int hight_binary_tree(Node*root)
76 {
77     int left,right;
78     if(root==NULL)
79     {
80         return 0;
81     }
82     else
83     {
84         left=hight_binary_tree(root->left);
85         right=hight_binary_tree(root->right);
86         if(left>right)
87         {
88             return left+1;
89         }
90         else
91         {
92             return left+1;
93         }
94     }
95     return 0;
96 }
97 int search(Node*root,int lkey)
98 {
99     //int lkey;
100    //int key;
101    Node *temp=root;
102    while(root!=NULL)
103    {
104        if(lkey == root->key)
105        {
106            return 1;
107        }
108        else if(lkey>root->key)
109        {
110            root=root->right;
111        }
112        else
113        {
114            root=root->left;
115        }
116    }
117    return 0;
118 }
119
120
121 int main()
122 {
123     Node*root=NULL;
124     int val,choice;
125     while(choice!=7)
126     {
127         printf("\n\nMenu\n");
128         printf("1.Binary tree insert\n");
129         printf("2.In-order display\n");
130         printf("3.Pre-order display\n");
131         printf("4.Post-order display\n");
132         printf("5.hight\n");

```

```

133 printf("6.search \n");
134 printf("7.Exit\n");
135 printf("\nEnter your choice\n");
136 scanf("%d",&choice);
137 switch(choice)
138 {
139     case 1:
140     {
141         printf("Enter data to insert\n");
142         scanf("%d",&val);
143         root=insert(root,val);
144         break;
145     }
146
147     case 2:
148     {
149         printf("In-order traversal\n");
150         inorder(root);
151         printf("\n");
152         break;
153     }
154
155     case 3:
156     {
157         printf("Pre-order traversal\n");
158         preorder(root);
159         printf("\n");
160         break;
161     }
162
163     case 4:
164     {
165         printf("Post-order traversal\n");
166         postorder(root);
167         printf("\n");
168         break;
169     }
170
171     case 5:
172     {
173         printf("hight of binary tree :");
174         printf("%d", hight_binary_tree(root));
175         break;
176     }
177     case 6:
178     {
179         int flag;
180         int skey;
181         printf("enter your key \n");
182         scanf("%d",skey);
183         flag=search(root,skey);
184         if(flag)
185         {
186             printf("key found \n");
187         }
188         else
189         {
190             printf("key not found \n");
191         }
192     }
193     break;
194
195
196     case 7:
197     {
198         printf("\nExiting the program\n");

```

```
199     break;
200 }
201
202
203
204 default:printf("\nInvalid choice\n");
205
206 }
207 }
208 return 0;
209 }
```