

```

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

```

```

67     postorder(root->right);
68     printf("%d\t",root->data);
69 }
70 }
71
72 int main()
73 {
74     Node*root=NULL;
75     int val,choice;
76     while(choice!=5)
77     {
78         printf("\n\nMenu\n");
79         printf("1.Binary tree insert\n");
80         printf("2.In-order display\n");
81         printf("3.Pre-order display\n");
82         printf("4.Post-order display\n");
83         printf("5.Exit\n");
84         printf("\nEnter your choice\n");
85         scanf("%d",&choice);
86         switch(choice)
87         {
88             case 1:
89             {
90                 printf("Enter data to insert\n");
91                 scanf("%d",&val);
92                 root=insert(root,val);
93                 break;
94             }
95
96             case 2:
97             {
98                 printf("In-order traversal\n");
99                 inorder(root);
100                printf("\n");
101                break;
102            }
103
104            case 3:
105            {
106                printf("Pre-order traversal\n");
107                preorder(root);
108                printf("\n");
109                break;
110            }
111
112            case 4:
113            {
114                printf("Post-order traversal\n");
115                postorder(root);
116                printf("\n");
117                break;
118            }
119
120            case 5:
121            {
122                printf("\nExiting the program\n");
123                break;
124            }
125
126            default:printf("\nInvalid choice\n");
127
128        }
129    }
130    return 0;
131 }

```