```c
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>

typedef struct treeNode
{
    int data;
    struct node*left;
    struct node*right;
    struct node*key;
}Node;

//Function to create a newnode
Node*createnode(int value)
{
    Node*newnode=(Node*)malloc(sizeof(Node));
    newnode->data=value;
    newnode->left=NULL;
    newnode->right=NULL;
    return newnode;
}

//Function to insert node in binary tree
Node*insert(Node*root,int value)
{
    if(root==NULL)
    {
        return createnode(value);
    }
    if(value<root->data)
    {
        root->left=insert(root->left,value);
    }
        else if(value>root->data)
        {
            root->right=insert(root->right,value);
        }
        return root;
}

//In-order traversal (LOR)
void inorder(Node *root)
{
    if(root!=NULL)
    {
        inorder(root->left);
        printf("%d\t",root->data);
        inorder(root->right);
    }
}

//Pre-order traversal (OLR)
void preorder(Node*root)
{
    if(root!=NULL)
    {
        printf("%d\t",root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

//Post-order traversal (LRO)
void postorder(Node*root)
{
    if(root!=NULL)
    {
        postorder(root->left);
        postorder(root->right);
        printf("%d\t",root->data);
    }
}


int hight_binary_tree(Node*root)
{
    int left,right;
    if(root==NULL)
    {
      return 0;
    }
    else
    {
        left=hight_binary_tree(root->left);
        right=hight_binary_tree(root->right);
        if(left>right)
        {
            return left+1;
        }
        else
        {
              return left+1;
        }
    }
    return 0;
}
int search(Node*root,int lkey)
{
    //int lkey;
    //int key;
    Node *temp=root;
    while(root=!NULL)
    {
        if(lkey == root->key)
        {
            return 1;
        }
        else if(lkey>root->key)
        {
            root=root->right;
        }
        else
        {
            root=root->left;
        }
    }
    return 0;
}


int main()
{
    Node*root=NULL;
    int val,choice;
while(choice!=7)
{
printf("\n\nMenu\n");
printf("1.Binary tree insert\n");
printf("2.In-order display\n");
printf("3.Pre-order display\n");
printf("4.Post-order display\n");
printf("5.hight\n");
printf("6.search \n");
printf("7.Exit\n");
printf("\nEnter your choice\n");
scanf("%d",&choice);
switch(choice)
{
    case 1:
    {
        printf("Enter data to insert\n");
        scanf("%d",&val);
        root=insert(root,val);
     break;
    }

    case 2:
    {
        printf("In-order traversal\n");
        inorder(root);
        printf("\n");
    break;
    }

    case 3:
    {
        printf("Pre-order traversal\n");
        preorder(root);
        printf("\n");
    break;
    }

    case 4:
    {
        printf("Post-order traversal\n");
        postorder(root);
        printf("\n");
    break;
    }

    case 5:
        {
            printf("hight of binary tree :");
            printf("%d", hight_binary_tree(root));
            break;
        }
    case 6:
        {
            int flag;
            int skey;
            printf("enter your key \n");
            scanf("%d",skey);
            flag=search(root,skey);
            if(flag)
            {
                printf("key found \n");
            }
            else
            {
                printf("key not found \n");
            }
        }
        break;


    case 7:
    {
        printf("\nExiting the program\n");
    break;
    }



default:printf("\nInvalid choice\n");

}
}
return 0;
}
```