```c
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *right;
    struct node *left;
};
struct node *getnode(int val)
{
    struct node *temp=(struct node*)malloc(sizeof(struct node));
    temp->right=NULL;
    temp->left=NULL;
    temp->data=val;
    return temp;
}
struct node *insert(struct node *root,int val)
{
    struct node *temp=getnode(val);
    if(root==NULL)
    return temp;
    struct node *cur=root;
    struct node *prev=NULL;
    while(cur!=NULL)
    {
        prev=cur;
        if(val<cur->data)
        cur=cur->left;
        else
        cur=cur->right;
    }
    if(val<prev->data)
    prev->left=temp;
    else
    prev->right=temp;
    return root;
}
void inorder(struct node *root)
{
    if(root==NULL)
    return;
    inorder(root->left);
    printf("%d ",root->data);
    inorder(root->right);
}
void postorder(struct node *root)
{
    if(root==NULL)
    return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ",root->data);
}
void preorder(struct node *root)
{
    if(root==NULL)
    return;
    printf("%d ",root->data);
    preorder(root->left);
    preorder(root->right);
}
int countnodes(struct node *root){
    if(root==NULL)
    return 0;
    return 1+countnodes(root->left)+countnodes(root->right);
}
void findmax(struct node *root)
{
    struct node *parent=NULL;
    struct node *cur=NULL;
    if(root==NULL)
    printf("root is empty\n");
    else
    {
        cur=root;
        while(cur->right!=NULL)
        {
            parent=cur;
            cur=cur->right;
        }
        if(parent==NULL){
        printf("parent node:NONE\n");
        printf("maximum node:%d\n",cur->data);
        return;
        }
        printf("maximum node: %d\n",cur->data);
        printf("parent node: %d\n",parent->data);
    }
}
void search(struct node *root,int val)
{
    if(root==NULL)
    {
        printf("tree is empty\n");
        return;
    }
    struct node *cur=root;
    struct node *parent=NULL;
    while(cur!=NULL)
    {
        if(val==cur->data)
        break;
        parent=cur;
        if(val<cur->data)
            cur=cur->left;
        else
            cur=cur->right;
    }
    if(cur==NULL)
    printf("key not found\n");
    else{
        if(parent==NULL){
            printf("parent node :NONE\n");
            printf("key %d found",cur->data);
        }
        else
        printf("parent node: %d\nkey %d found\n",parent->data,cur->data);
    }
}
int max(int a,int b)
{
    return (a>b)?a:b;
}
int height(struct node *root)
{
    if(root==NULL)
    return -1;
    else
    {
        int leftheight=height(root->left);
        int rightheight=height(root->right);
        return 1+max(leftheight,rightheight);
    }
}
void main()
{
    int choice,val;
    struct node *root=NULL;
    printf("main
menu\n1.insert\n2.inorder\n3.postorder\n4.preorder\n5.count nodes\n6.find
max\n7.search\n8.exit\n");
    for(;;)
    {
    printf("\nenter your choice: ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:printf("enter the value:");
```

```
                scanf("%d",&val);
                root=insert(root,val);
                break;
        case 2:inorder(root);
                break;
        case 3:postorder(root);
                break;
        case 4:preorder(root);
                break;
        case 5:printf(" number of node is: %d",countnodes(root));
                break;
        case 6:findmax(root);
                break;
        case 7:printf("enter the key to search: ");
                scanf("%d",&val);
                search(root,val);
                break;
        case 8:printf("height of the tree: %d",height(root));
                break;
        case 9:printf("thank you!!");
                exit(0);
        default:printf("invalid choice\n");

    }
    }
}
```