Binary Search Tree

```
#include<iostream.h>
#include<conio.h>
class bst
{
        int data;
        bst *left;
        bst *right;
        public:
        void add(int,bst **);
        void preorder(bst *);
        void inorder(bst *);
        void postorder(bst *);
        int search(int,bst **);
        void min(bst **);
        void max(bst **);
        void count(bst **);
};
bst *root,*temproot;
void bst::preorder(bst *q)
{
        if(q!=NULL)
        {
                cout<<q->data<<"\t";
                preorder(q->left);
                preorder(q->right);
        }
}
void bst::inorder(bst *q)
{
        if(q!=NULL)
        {
                inorder(q->left);
                cout<<q->data<<"\t";
                inorder(q->right);
        }
}
void bst::postorder(bst *q)
{
        if(q!=NULL)
        {
                postorder(q->left);
                postorder(q->right);
                cout<<q->data<<"\t";
        }
}
void bst::add(int num,bst **q)
{
        if((*q)==NULL)
        {
                (*q)=new bst;
```

```
                (*q)->data=num;
                (*q)->left=NULL;
                (*q)->right=NULL;
        }
        else
        {
                if(num<(*q)->data)
                        add(num,&(*q)->left);
                else
                        add(num,&(*q)->right);
        }
}
int bst::search(int num,bst **q)
{
        if((*q)==NULL)
                return -1;
        if((*q)->data==num)
                return 1;
        else
        {
                if(num<(*q)->data)
                        search(num,&(*q)->left);
                else if (num>(*q)->data)
                        search(num,&(*q)->right);
        }
}
void bst::min(bst **q)
{
        int mn=root->data;
        while((*q)!=NULL)
        {
                if(mn>(*q)->data)
                        mn=(*q)->data;
                (*q)=(*q)->left;
        }
        cout<<mn<<"is the smallest values";
}
void bst::max(bst **q)
{
        int mx=root->data;
        while((*q)!=NULL)
        {
                if(mx<(*q)->data)
                        mx=(*q)->data;
                (*q)=(*q)->right;
        }
        cout<<mx<<"is the greatest values";
}
void bst::count(bst **q)
{
        int cot=0;
```

```
	bst *temp=(*q);
	if(root==NULL)
		cout<<"Number of node=0";
	else
	{
		while(temp!=NULL)
		{
			cot++;
			temp=temp->left;
		}
		temp=(*q)->right;
		while(temp!=NULL)
		{
			cot++;
			temp=temp->right;
		}
		cout<<"\nCount of nodes= "<<cot;
	}
}
void main()
{
	clrscr();
	bst b;
	int num,opt;
	char ch='y';
	root=NULL;
	while(ch=='y')
	{

	cout<<"\n1.add\n2.preorder\n3.inorder\n4.postorder\n5.search\n6.min\n7.max\n8.count\
n";
		cout<<"\nEnter option you want to perform: ";
		cin>>opt;
		switch(opt)
		{
			case 1: cout<<"\nEnter element to add: ";
				cin>>num;
				b.add(num,&root);
				break;
			case 2: temproot=root;
				b.preorder(temproot);
				break;
			case 3: temproot=root;
				b.inorder(temproot);
				break;
			case 4: temproot=root;
				b.postorder(temproot);
				break;
			case 5: cout<<"\nEnter element to search: ";
				cin>>num;
				temproot=root;
```

```
                                int result=b.search(num,&temproot);
                                if(result==1)
                                        cout<<"Element Found";
                                else
                                        cout<<"Element Not found";
                                break;
                        case 6: temproot=root;
                                b.min(&temproot);
                                break;
                        case 7: temproot=root;
                                b.max(&temproot);
                                break;
                        case 8: b.count(&root);
                                break;
                }
                cout<<"\nDo you want to CONTINUE ?(y/n): ";
                cin>>ch;
        }
        getch();
}
```