Assignment no 4

Input

```
#include<iostream>
using namespace std;
class node
{
    public:
           int data;
           node *left;
           node *right;
};
class bst
{

public:
                      node *root;
           bst()
           {
                      root=NULL;
           }
           void create();
           void insert();
           void postorder(node*);
           void inorder(node *);
       void preorder(node *);
       void search(int key);
      // int search(node*, int key);
           void minimum();
           int height(node*);
};


void bst::minimum()
{
      node *temp;
           int min;
     temp=root;
                while(temp->left!=NULL)
                   {
                min=temp->data;
                               temp=temp->left;
                               if(temp->data<min)
                               {
                                          min=temp->data;
                               }
                               else
```

```
            {
                temp=temp->left;

            }
        }
        cout<<"minimum no. is:"<<min;
}
int bst::height(node *root)
{
    if(root==NULL)
    {
        return 0;
    }
    else
    {
        if(height(root->right)>height(root->left))                  //right tree is longer
        {
            return (1+height(root->right));
        }
        else
        {
            return (1+height(root->left));
        }
    }
}

void bst::create()
{
    node *curr,*temp;
    int ans=1;
    cout<<"enter data:";
    do
    {
        curr=new node;
        cin>>curr->data;
        curr->left=curr->right=NULL;
        if(root==NULL)
        {
            root=curr;
        }
        else
        {
            temp=root;
            while(1)
            {
                if(curr->data<=temp->data)
                {
                    if(temp->left==NULL)
                    {
                        temp->left=curr;
                        break;
                    }
                    else
                    {
                        temp=temp->left;
                    }
                }
                else
                {
                    if(temp->right==NULL)
                    {
                        temp->right=curr;
                        break;
                    }
                    else
                    {
                        temp=temp->right;
                    }
                }
            }
        }
        cout<<"want to continue:";
        cin>>ans;
    }while(ans==1);
}
void bst::inorder(node *root)
{
    if(root!=NULL)
    {
        inorder(root->left);
        cout<<" "<<root->data;
        inorder(root->right);
    }
}
void bst::preorder(node *root)
{
    if(root!=NULL)
    {
        cout<<"  "<<root->data;
        preorder(root->left);
        preorder(root->right);
    }
```

```
}
void bst::postorder(node *root)
{
        if(root!=NULL)
        {
                postorder(root->left);
                postorder(root->right);
                cout<<" "<<root->data;
        }
}
void bst::insert()
{
        node *curr,*temp;
        int ans=1;
        cout<<"enter data:";


                curr=new node;
                cin>>curr->data;
                curr->left=curr->right=NULL;
                if(root==NULL)
                {
                        root=curr;
                }
                else
                {
                        temp=root;
                        while(1)
                        {
                                if(curr->data<=temp->data)
                                {
                                        if(temp->left==NULL)
                                        {
                                                temp->left=curr;
                                                break;
                                        }
                                        else
                                        {
                                                temp=temp->left;
                                        }
                                }
                                else
                                {
                                        if(temp->right==NULL)
                                        {
                                                temp->right=curr;
                        break;
                    }
                    else
                    {
                        temp=temp->right;
                    }
                }
            }//end of while
        }
}


void bst::search(int key)
{
    node *curr;
    curr=root;
    while(curr!=NULL)
    {
        if(curr->data==key)
        {
            cout<<"found";
            break;
        }
        else
        {
            if(key<curr->data)
            {
                curr=curr->left;
            }
            else
            {
                curr=curr->right;
            }
        }
    }
    if(curr==NULL)      //not found even at the end of the tree.
    {
        cout<<"not found";
    }
}

int main()
{
    bst b;
    int key,ch;
    do
```

```
	{
	cout<<"\n1.create\n2.insert\n3.inorder\n4.preorder\n5.postorder\n6.search\n7.minimu
m\n8.height\npress 0 to exit\n";
		cout<<"enter your choice:";
		cin>>ch;
		switch(ch)
		{
			case 1:b.create();
				break;
			case 2:b.insert();
				break;
			case 3:cout<<"inorder traversal is\n";
					b.inorder(b.root);
					break;
			case 4:cout<<"preorder traversal is\n";
					b.preorder(b.root);
					break;
			case 5:cout<<"postorder traversal is\n";
					b.postorder(b.root);
					break;
			case 6:cout<<"\nenter key:";
				cin>>key;
				b.search(key);
				break;
			case 7:b.minimum();
				break;
			case 8:cout<<"height of tree: "<<b.height(b.root);
				break;
		}
	}while(ch!=0);
	return 0;
}
```

Output

```
1.create
2.insert
3.inorder
4.preorder
5.postorder
6.search
7.minimum
8.height
press 0 to exit
enter your choice:1
enter data:58
```

```
want to continue:1
2
want to continue:1
47
want to continue:1
69
want to continue:1
24
want to continue:0
1.create
2.insert
3.inorder
4.preorder
5.postorder
6.search
7.minimum
8.height
press 0 to exit
enter your choice:2
enter data:61
1.create
2.insert
3.inorder
4.preorder
5.postorder
6.search
7.minimum
8.height
press 0 to exit
enter your choice:3
inorder traversal is
 2 24 47 58 61 69
1.create
2.insert
3.inorder
4.preorder
5.postorder
6.search
7.minimum
8.height
press 0 to exit
enter your choice:4
preorder traversal is
  58  2  47  24  69  61
1.create
2.insert
```

```
3.inorder
4.preorder
5.postorder
6.search
7.minimum
8.height
press 0 to exit
enter your choice:5
postorder traversal is
 24 47 2 61 69 58
1.create
2.insert
3.inorder
4.preorder
5.postorder
6.search
7.minimum
8.height
press 0 to exit
enter your choice:6
enter key:61
found
1.create
2.insert
3.inorder
4.preorder
5.postorder
6.search
7.minimum
8.height
press 0 to exit
enter your choice:7
minimum no. is:2
1.create
2.insert
3.inorder
4.preorder
5.postorder
6.search
7.minimum
8.height
press 0 to exit
enter your choice:8
height of tree: 4
1.create
2.insert
```

```
3.inorder
4.preorder
5.postorder
6.search
7.minimum
8.height
press 0 to exit
enter your choice:0
--------------------------------
Process exited after 89.52 seconds with return value 0
Press any key to continue . . .
```