Binary Tree Class

```
#include <iostream>
#include<stdio.h>
#include "QueueCpp.h"

using namespace std;
class Tree
{
    Node *root;

public:
    Tree(){root=NULL;}
    void CreateTree();
    void Preorder(){Preorder(root);}
    void Preorder(Node *p);
    void Postorder(){Postorder(root);}
    void Postorder(Node *p);
    void Inorder(){Inorder(root);}
    void Inorder(Node *p);
    void Levelorder(){Levelorder(root);}
    void Levelorder(Node *p);
    int Height(){return Height(root);}
    int Height(Node *root);
};

void Tree::CreateTree()
{
    Node *p,*t;
    int x;
    Queue q(100);

    printf("Eneter root value ");
    scanf("%d",&x);
    root=new Node;
    root->data=x;
    root->lchild=root->rchild=NULL;
    q.enqueue(root);
```

```
    while(!q.isEmpty())
    {
        p=q.dequeue();
        printf("eneter left child of %d ",p->data);
        scanf("%d",&x);
        if(x!=-1)
        {
            t=new Node;
            t->data=x;
            t->lchild=t->rchild=NULL;
            p->lchild=t;
            q.enqueue(t);
        }
        printf("eneter right child of %d ",p->data);
        scanf("%d",&x);
        if(x!=-1)
        {
            t=new Node;
            t->data=x;
            t->lchild=t->rchild=NULL;
            p->rchild=t;
            q.enqueue(t);
        }
    }
}

void Tree::Preorder(struct Node *p)
{
    if(p)
    {
        printf("%d ",p->data);
        Preorder(p->lchild);
        Preorder(p->rchild);
    }
}

void Tree::Inorder(struct Node *p)
{
    if(p)
    {
```

```
        Inorder(p->lchild);
        printf("%d ",p->data);
        Inorder(p->rchild);
    }
}
void Tree::Postorder(struct Node *p)
{
    if(p)
    {
        Postorder(p->lchild);
        Postorder(p->rchild);
        printf("%d ",p->data);
    }
}

void Tree::Levelorder(struct Node *root)
{
    Queue q(100);

    printf("%d ",root->data);
    q.enqueue(root);

    while(!q.isEmpty())
    {
        root=q.dequeue();
        if(root->lchild)
        {
            printf("%d ",root->lchild->data);
            q.enqueue(root->lchild);
        }
        if(root->rchild)
        {
            printf("%d ",root->rchild->data);
            q.enqueue(root->rchild);
        }
    }
}

int Tree::Height(struct Node *root)
{
```

```
    int x=0,y=0;
    if(root==0)
        return 0;
    x=Height(root->lchild);
    y=Height(root->rchild);
    if(x>y)
        return x+1;
    else
        return y+1;

}


int main()
{
    Tree t;
    t.CreateTree();
    cout<<"Preorder ";
    t.Preorder();
    cout<<endl;
    cout<<"Inorder ";
    t.Inorder();
    cout<<endl<<endl;

    return 0;
}
```