Given a non-empty binary tree, find the maximum path sum. A path is defined as any sequence of nodes from some starting node to any node in the tree along the parent-child connections. The path must contain at least one node and does not need to go through the root.The function should return an integer representing the maximum path sum.

#include <bits/stdc++.h>

#include <queue>

using namespace std;

class TreeNode{

public:

int data;

TreeNode\* right;

TreeNode\* left;

TreeNode(int data){

this->data=data;

this->left=NULL;

this->right=NULL;

}

};

// creating tree using level order traversal

TreeNode\* buildTree(TreeNode\* &root){

cout<<"enter the root data"<<endl;

int data;

cin>>data;

root =new TreeNode(data);

queue<TreeNode\*> q;

q.push(root);

while(!q.empty()){

TreeNode\* front=q.front();

q.pop();

cout<<"enter the left of: "<<front->data<<endl;

int leftData;

cin>>leftData;

if(leftData!=-1){

front->left=new TreeNode(leftData);

q.push(front->left);

}

cout<<"enter the right of: "<<front->data<<endl;

int rightData;

cin>>rightData;

if(rightData!=-1){

front->right=new TreeNode(rightData);

q.push(front->right);

}

}

return root;

}

void printLevelOrder(TreeNode\* root){

queue<TreeNode\*> q;

q.push(root);

q.push(NULL);

while(!q.empty()){

TreeNode\* front= q.front();

q.pop();

if(front==NULL){

cout<<endl;

if(!q.empty()) {

q.push(NULL);

}

continue;

}

cout<<front->data<<" ";

if(front->left)

q.push(front->left);

if(front->right)

q.push(front->right);

}

}

// -10 5 6 7 8 9 -1 11 -1 -1 -1 -1 -1 -1 -1

int maxPathSum(TreeNode\* &root, int& sum){

if(root==NULL){

return 0;

}

int leftSum=maxPathSum(root->left,sum);

int rightSum=maxPathSum(root->right,sum);

sum=max(sum,root->data+rightSum+leftSum);

return max(0,root->data+max(leftSum,rightSum));

}

int main(){

TreeNode\* root = NULL;

root = buildTree(root);

//printing the list level order

printLevelOrder(root);

//finding maximum path sum

int sum=INT\_MIN;

maxPathSum(root,sum);

cout<<"max sum is: "<<sum<<endl;

}

Output

