Convert tree into linked list so that output is in increasing order

#include <iostream>

#include<queue>

using namespace std;

class node{

public:

int data;

node \*left;

node \*right;

node(int d){

data=d;

left=NULL;

right=NULL;

}

};

node\* insertInBST(node \*root,int data){

if(root==NULL){

return new node(data);

}

if(data<=root->data){

root->left=insertInBST(root->left,data);

}

else{

root->right=insertInBST(root->right,data);

}

return root;

}

node\* build(){

int d;

cin>>d;

node \*root=NULL;

while(d!=-1){

root=insertInBST(root,d);

cin>>d;

}

return root;

}

void inorder(node \*root){

if(root==NULL)

return;

inorder(root->left);

cout<<root->data<<", ";

inorder(root->right);

}

void bfs(node \*root){

queue<node\*> q;

q.push(root);

q.push(NULL);

while(!q.empty()){

node\* f = q.front();

if(f==NULL){

cout<<endl;

q.pop();

if(!q.empty()){

q.push(NULL);

}

}

else{

cout<<f->data<<",";

q.pop();

if(f->left){

q.push(f->left);

}

if(f->right){

q.push(f->right);

}

}

}

return;

}

class LinkedList{

public:

node \*head;

node \*tail;

};

LinkedList flatten(node \*root){

LinkedList l;

if(root==NULL){

l.head = l.tail = NULL;

return l;

}

//leaf node

if(root->left==NULL && root->right==NULL){

l.head=l.tail=root;

return l;

}

// Left is not null

if(root->left!=NULL && root->right==NULL){

LinkedList leftLL=flatten(root->left);

leftLL.tail->right=root;

l.head=leftLL.head;

l.tail=root;

return l;

}

// Right is not null

if(root->left==NULL && root->right!=NULL){

LinkedList rightLL=flatten(root->right);

root->right=rightLL.head;

l.head=root;

l.tail=rightLL.tail;

return l;

}

// Both sides are not null

LinkedList leftLL=flatten(root->left);

LinkedList rightLL=flatten(root->right);

leftLL.tail->right=root;

root->right=rightLL.head;

l.head=leftLL.head;

l.tail=rightLL.tail;

return l;

}

int main()

{

node \*root=build();

inorder(root);

cout<<endl;

bfs(root);

cout<<endl;

LinkedList l=flatten(root);

node \*temp=l.head;

while(temp!=NULL){

cout<<temp->data<<" -->";

temp=temp->right;

}

}

Input-

5 3 7 1 6 8 -1

Output-

1, 3, 5, 6, 7, 8,

5,

3,7,

1,6,8,

1 -->3 -->5 -->6 -->7 -->8 -->