1. Problem, find all the nodes value in between the root node and given value.

Explanation:

1

T / \

2 3

F / \ T

4 5

F/ \F F/ \T

6 7

And X=7, root Node.

Vector<int> 1 ,2 5 7

Check condition 1!=7, 2!=7 4!=7 5!=7 6!=7 7==7 return 0r break the function,

Left 2

Right 3 5 7

Code:

Vector<int> getelement(Node\* root, int x)

{

Vector<int> ans;

Getallelement(root, ans,x);

Return ans;   
}

Bool Getallelement( Node \*root,vector<int>& ans,int x)

{

If(root==NULL)return false;

ans.push\_back(root->val);

if(root->val==X)return true;

if( Getallelement(root->left, ans,x)| Getallelement(root->right, ans,x))

{ return true;}

Ans.pop\_back();

Return false;

}

Space complexity: O(N) ( answer vector) + O(H) (recursive call)

Time complexity: O(N) Good ecplanatation:

Question said that return all the possible node value sum of root to any is equal to target.

Code: some changes as you can see

void Getallelement(Node \*root,vector<int>& v,int sum, vector<vector<int>>&ans,int targetsum)

{

if(root==NULL)return;

v.push\_back(root->key);

targetsum+=root->key;

if(sum==targetsum)

{

ans.push\_back(v);

}

if(root->left!=NULL)

{

Getallelement(root->left,v,sum,ans,targetsum);

}

if(root->right!=NULL)

{

Getallelement(root->right,v,sum,ans,targetsum);

}

v.pop\_back();

}

vector<vector<int>> printPaths(Node \*root, int sum)

{

//code here

vector<vector<int>>ans;

vector<int>v;

if(root==NULL)return ans;

Getallelement(root,v,sum,ans,0);

return ans;

}

=================================================================================

1. Find the Sum of Root to leaf is equal to target. If found return true else return false.

Explanation:

10

/ \

8 2 target=23;

/ \ /

3 5 2

Stack Node 1: 10p 8p 2 3 5

Stack sum 2: 10 18 12 21 23found return;

Code:

bool sumofRoot(Node \*root, int target)

{

stack<Node\*> s1;

stack<int> s1;

int sum=0;

s1.push(root);

s2.push(root->val);

while(!s1.empty())

{

root =s1.pop();

s1.pop();

sum= s2.top();

s2.pop();

if(sum==target&&root->left==NULL&&root->right==NULL)return true;

if(root->left!=NULL){s1.push(root->left);s2.push(root->left->val+sum);}

if(root->right!=NULL){s1.push(root->right);s2.push(root->right->val+sum);}

}

return false;  
}

Space complexity: O(N+N)

Time complexity: O(N)