Assignment Solutions | Binary Trees 3 | Week 17

1. Left View of Binary Tree

    Given a Binary Tree, print the Left view of it.

    The left view of a binary tree refers to the set of nodes that are visible when the tree is viewed from the left side.

   Sol:

```
class Solution {
public:
    void helper(TreeNode *root, int level, vector<int> &ans){
        if(root==NULL) return ;
        if(ans.size()==level) ans.push_back(root->val);
        helper(root->left, level+1, ans);
        helper(root->right, level+1, ans);
    }
    vector<int> leftSideView(TreeNode *root) {
        vector<int> ans;
        helper(root, 0, ans);
        return ans;
    }
};
```

2. Path Sum I [LeetCode 112]

   Sol:

```
class Solution {
public:
    bool hasPathSum(TreeNode* root, int sum) {
        if(root == NULL){
            return false;
        }
        int newsum = sum - root->val;
        if(root->left == NULL && root->right == NULL){
            return newsum == 0;
        }
        return hasPathSum(root->left, newsum) || hasPathSum(root->right, newsum);
    }
};
```

3. Construct Binary Tree from Inorder & Postorder   Traversal [LeetCode 106]

Sol:

```
class Solution {
public:
    TreeNode* build(vector<int>& in, int inLo, int inHi, vector<int>& post, int
postLo, int postHi){
        if(inLo > inHi) return NULL;
        TreeNode* root = new TreeNode(post[postHi]);
        if(inLo == inHi) return root;
        int i = inLo;
        while(i<inHi){
            if(in[i] == post[postHi]) break;
            i++;
        }
        int leftcount = i - inLo; int rightcount = inHi - i;
        root->left = build(in,inLo,i-1,post,postLo,postLo+leftcount-1);
        root->right = build(in,i+1,inHi,post,postLo+leftcount,postHi-1);
        return root;
    }
    TreeNode* buildTree(vector<int>& in, vector<int>& post) {
        int n = in.size();
        return build(in,0,n-1,post,0,n-1);
    }
};
```

4.Construct Binary Tree from Preorder & Postorder Traversal [LeetCode 889]

Sol :

```
class Solution {
public:
    TreeNode* dfs(vector<int>& preorder,int prestart,int preend, vector<int>&
postorder,int poststart,int postend){
        if(prestart>preend) return NULL;
        if(poststart>postend) return NULL;
        TreeNode* root=new TreeNode(preorder[prestart]);
        if(prestart==preend) return root;
        int postindex=poststart;
        while(postorder[postindex]!=preorder[prestart+1]){
            postindex++;
        }
        int len= postindex-poststart+1;
        root-
>left=dfs(preorder,prestart+1,prestart+len,postorder,poststart,postindex);
        root-
>right=dfs(preorder,prestart+len+1,preend,postorder,postindex+1,postend-1);
        return root;
    }
    TreeNode* constructFromPrePost(vector<int>& preorder, vector<int>& postorder) {
        return dfs(preorder,0,preorder.size()-1,postorder,0,postorder.size()-1);
    }
};
```

Note:- Please try to invest time doing the assignments which are necessary to build a strong foundation. Do not directly Copy Paste using Google or ChatGPT. Please use your brain😀.
