Binary Trees
(Assignment Solutions)

Question 1 :

```
bool isUnivalTree(TreeNode* root) {
        if(root == NULL) {
            return true;
        }

        if(!isUnivalTree(root->left) || !isUnivalTree(root->right)) {
            return false;
        }

        if(root->left != NULL) {
            if(root->val != root->left->val) {
                return false;
            }
        }

         if(root->right != NULL) {
            if(root->val != root->right->val) {
                return false;
            }
        }

        return true;
    }
```

Question 2 :

```
TreeNode* invertTree(TreeNode* root) {
        if(root == NULL) {
            return root;
        }

        TreeNode* left = invertTree(root->left);
        TreeNode* right = invertTree(root->right);
```

```
        root->left = right;
        root->right = left;
        return root;
    }
```

Question 3 :

```
TreeNode* removeLeafNodes(TreeNode* root, int target) {
        if(root == NULL) {
            return NULL;
        }

        TreeNode* left = removeLeafNodes(root->left, target);
        TreeNode* right = removeLeafNodes(root->right, target);

        if(left == NULL && right == NULL && root->val == target) {
            return NULL;
        }

        root->left = left;
        root->right = right;
        return root;
    }
```

Question 4 :

```
string duplicate(TreeNode* root,unordered_map<string,int>
&mp,vector<TreeNode*> &v){

    if(root==NULL) return "";
    string a=duplicate(root->left,mp,v);
    string b=duplicate(root->right,mp,v);
    string temp=to_string(root->val)+","+a+","+b;
    mp[temp]++;
    if(mp[temp]==2) v.push_back(root);
    return temp;
```

```
}
    vector<TreeNode*> findDuplicateSubtrees(TreeNode* root) {
        unordered_map<string,int> mp;
        vector<TreeNode*> v;
        duplicate(root,mp,v);
        return v;
    }
```

Question 5 :

```
int height(TreeNode *root,int &maxi){
        if(root==NULL){
            return 0;
        }

        int lh = max(0,height(root->left,maxi));
        int rh = max(0,height(root->right,maxi));

        maxi = max(maxi,lh+rh+root->val);
        return root->val+max(lh,rh);
    }
    int maxPathSum(TreeNode* root) {
        int res=INT_MIN;
        height(root,res);
        return res;
    }
```

https://telegram.me/+QGUyTripaP4zNTcx

https://telegram.me/+nEKeBr_yhXtmY2Yx

Say Hi to TG for Further Updates:
https://t.me/RepublicDayBot

#TGSFamily