Balanced Height (height<=1)

Approach 1 (using pair class) O(n)

Code:

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    bool isBalanced(TreeNode* root) {
        //using pair
        pair<int,bool> fi= ans(root);
        return fi.second;

    }
    //calculate hieght of a node
    int hieght(TreeNode* root)
    {
        if(root==nullptr)
            return 0;
        return 1+max(hieght(root->left),hieght(root->right));
    }

   /* using pair class becuase each node maintain hieght and balanced factor
    so each node saperately store two values 1. hieght 2.bool (subtree is balanced or not)*/

    // fuction definition of ans
    pair<int,bool> ans(TreeNode* root)
    {
        if(root==NULL)
        {
            pair<int,bool> p;
            p.first=0;
            p.second=true;
            return p;
```

```
        }
        pair<int,bool> temp;
        temp.first=hieght(root);
        temp.second=true;
        pair<int,bool> leftAns=ans(root->left);
        pair<int,bool> rightAns=ans(root->right);

        int leftHieght=1+max(leftAns.first,rightAns.first);
        bool trueOrFalse;
      if(abs(leftAns.first-rightAns.first)<=1)
      {
         trueOrFalse=leftAns.second && rightAns.second;
      }
        else
        {trueOrFalse=false;}

        pair<int,bool> main;
        main.first=leftHieght;
        main.second=trueOrFalse;
        return main;


    }
};
```

Approach 2 using simple O(n)

Code:

```
Class
{
public:
bool isBalanced(BinaryTreeNode<int> *root) {
     if (root == NULL)
       return 1;
   int l=height(root->left);
         int r=height(root->right);

    if (  abs(l- r)<=1 && isBalanced(root->left) && isBalanced(root->right))
       return 1;

  return 0;
}
 int hieght(TreeNode* root)
   {
       if(root==nullptr)
```

```
        return 0;
    return 1+max(hieght(root->left),hieght(root->right));
  }
};
```