给定一个二叉树，根节点为第1层，深度为 1。在其第 `d` 层追加一行值为 `v` 的节点。

添加规则：给定一个深度值 `d` （正整数），针对深度为 `d-1` 层的每一**非空**节点 `N`，为 `N` 创建两个值为 `v` 的左子树和右子树。

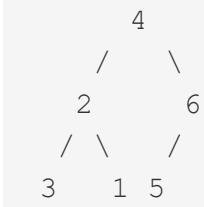
将 `N` 原先的左子树，连接为新节点 `v` 的左子树；将 `N` 原先的右子树，连接为新节点 `v` 的右子树。

如果 `d` 的值为 1，深度 d - 1 不存在，则创建一个新的根节点 `v`，原先的整棵树将作为 `v` 的左子树。

示例 1:

输入：

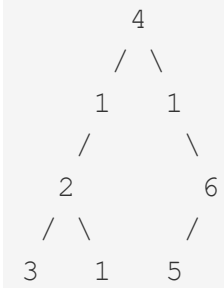
二叉树如下所示：



v = 1

d = 2

输出：



示例 2:

输入：

二叉树如下所示：

v = 1

d = 3

输出：

```
       4
      /
     2
    / \
   1   1
  /     \
 3       1
```

注意:

1. 输入的深度值 d 的范围是：[1，二叉树最大深度 + 1]。
2. 输入的二叉树至少有一个节点。

思路，记录遍历的深度，将度为d-1的节点，所连接的节点都改为需要插入节点的拼接即可。把度为1的节点当作特殊节点剔除（这里需注意数据初始化，struct中包含对应的构造函数，和类的使用基本一致）

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    void dfs(TreeNode* root, int v, int d,int wz){
        if(wz<d){
            if(root->left!=NULL){
                dfs(root->left,v,d,wz+1);
            }
            if(root->right!=NULL){
                dfs(root->right,v,d,wz+1);
            }
        }else{
```

```
            TreeNode *node=new TreeNode(v);
            node->left=root->left;
            root->left=node;
            TreeNode *node1=new TreeNode(v);
            node1->right=root->right;
            root->right=node1;
        }
    }
    TreeNode* addOneRow(TreeNode* root, int v, int d) {
        if(d==1){
            TreeNode *node=new TreeNode(v);
            node->left=root;
            return node;
        }
        dfs(root,v,d,2);
        return root;
    }
};
```