8.2

6.49

```
#include <algorithm>
#include <iostream>
#include <queue>
#include <vector>

using namespace std;

typedef struct TreeNode
{
    int val;
    TreeNode *left;
    TreeNode *right;
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
} Tree;

bool Judge(Tree *tree)
{
    if (tree == nullptr)
        return 0;
    queue<Tree *> t;
    t.push(tree);
    int cur_wide = 1;
    int flag = 0; // 标记树是否应该再无后继节点
    while (!t.empty())
    {
        for (int i = 0; i < static_cast<int>(t.size()); i++)
        {
            Tree *child = t.front();
            t.pop();
            if (flag)
            {
                if (child->left != nullptr || child->right != nullptr)
                    return false;
            }
            else if (child->left == nullptr && child->right != nullptr)
                return false;
            else if (child->left == nullptr && child->right != nullptr)
                flag = 1;
            else
            {
                t.push(child->left);
                t.push(child->right);
            }
        }
    }
```

```
    return true;
}
```

6.69

```
#include <algorithm>
#include <iostream>
#include <stack>
#include <vector>

using namespace std;

typedef struct TreeNode
{
    char str;
    TreeNode *left;
    TreeNode *right;
    TreeNode(int x) : str(x), left(nullptr), right(nullptr) {}
} Tree;

stack<char> str_s;
stack<int> space_s;

void IOT(Tree *tree, int floor)
{
    if (tree)
    {
        IOT(tree->left, floor + 1);
        str_s.push(tree->str);
        space_s.push(floor);
        IOT(tree->right, floor + 1);
    }
}

void PrintTree(Tree *tree)
{
    IOT(tree, 0);
    int space = space_s.top();
    char str = str_s.top();
    for (int i = 0; i < space; i++)
        cout << " " << "";
    cout << str << endl;
    space_s.pop();
    str_s.pop();
}
```