340 Lab - Multiple Functions for Binary Tree Comparison

You will write several functions to compare two binary trees:

1. If the two binary trees are the same? -- Two binary trees are considered the same if they are structurally identical, and the nodes have the same value. If both are empty trees, return true.
2. If the two binary trees are symmetric? -- They need to have mirrored tree structure, as well as values.
3. If one binary tree is a subtree of another tree? – Return true if the first tree is part of the second tree. Note that different nodes can have the same value in a tree. If they are the same tree, return false, but in the case that both are empty trees, return false.
4. If the two binary trees are the same heap? – Return tree if the two trees are the same and they are both a max heap. Empty tree is considered a heap.
5. If the two binary trees have the same width? -- The width of binary tree is the maximum width of all levels. The width of one level is defined as the length between the end-nodes (the leftmost and rightmost non-null nodes). Hint: You can use level order traversal -- You can use a queue to traverse the binary tree level by level, keeping track of the position of each node. The width of an empty free is defined as 0.

Since you work in a group, you can divide the functions among group members. #5 may take a bit longer to code than others.

Input: Two binary trees (Visualization provided to you).
Output: A boolean value from each function.

Your tasks:
Given the roots of two binary trees p and q, write the following functions, and put in solution.cpp.

```
bool isSameTree(TreeNode* p, TreeNode* q);
bool isSymmetricTree(TreeNode* p, TreeNode* q);
bool isOneTreePartOfAnother(TreeNode* p, TreeNode* q);
bool isSameHeap(TreeNode* p, TreeNode* q);
bool ofSameWidth(TreeNode* p, TreeNode* q);
```

Definition for a binary tree node in a header file *treenode.h*:

```
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) {}
};
```

Examples:

```
│   ┌── 3
└── 1
    └── 2
        └── 4
│   ┌── 3
└── 1
    └── 2
        └── 4

isSameTree? 1
isSymmetricTree? 0
isOneTreePartOfAnother? 1
isSameHeap? 0
ofSameWidth? 1


│   ┌── 4
└── 1
    └── 2
│   ┌── 3
└── 1
    └── 2
        └── 4

isSameTree? 0
isSymmetricTree? 0
isOneTreePartOfAnother? 0
isSameHeap? 0
ofSameWidth? 1
│   ┌── 1
└── 1
    └── 2
│   ┌── 1
└── 1
isSameTree? 0
isSymmetricTree? 0
isOneTreePartOfAnother? 0
isSameHeap? 0
ofSameWidth? 0


└── 1
    └── 2
│   ┌── 2
└── 1
isSameTree? 0
isSymmetricTree? 1
isOneTreePartOfAnother? 0
isSameHeap? 0
ofSameWidth? 1
```

Additional notes

If you will write your own main function to test the output for various test cases, you need to write your own insert method to build a binary tree, which you should have done in assignments.