Same Binary Tree [LeetCode](https://leetcode.com/problems/same-tree/description/)

Two binary trees are considered the same if they are structurally identical, and the nodes have the same value.

**Approach 1: Recursive function to check if two binary trees are the same**

* In the **isSameTreeRecursively** function, you check if two binary trees are the same using a recursive approach.
* If both trees are empty, they are considered the same.
* If one tree is empty while the other is not, they are not the same.
* If both trees have nodes at the same position with the same values, you recursively check their left and right subtrees.
* If all conditions are met, you return **true**; otherwise, you return **false**.

**Time Complexity:** **O(N), where N is the number of nodes in the binary tree. You visit each node once.**

**Space Complexity: O(H), where H is the height of the binary tree due to the function call stack.**

**Approach 2: Iterative function to check if two binary trees are the same**

* In the **isSameTreeIterative** function, you check if two binary trees are the same using an iterative approach.
* If both trees are empty, they are considered the same.
* If one tree is empty while the other is not, they are not the same.
* You use two stacks (**st1** and **st2**) to perform iterative traversal of both trees.
* You push the root nodes of both trees onto the stacks.
* While both stacks are not empty, you compare nodes from both trees.
* If nodes have different values or one node is empty while the other is not, they are not the same.
* You push the left and right children of both nodes onto the stacks for further comparison.
* If both stacks become empty, all nodes have been compared and found to be the same.

**Time Complexity: O(N), where N is the number of nodes in the binary tree. You visit each node once.**

**Space Complexity: O(H), where H is the height of the binary tree due to the two stacks.**