Our Solution(s) Run Code

Solution 1 Solution 2

```swift
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class Program {
    class BinaryTree {
        var value: Int
        var left: BinaryTree?
        var right: BinaryTree?

        init(value: Int) {
            self.value = value
            left = nil
            right = nil
        }
    }

    // O(n) time | O(d) space - where n is the number of nodes in the
    // and d is the depth (height) of the Binary Tree
    func flattenBinaryTree(root: BinaryTree) -> BinaryTree {
        var result = flattenTree(node: root)
        return result.leftMost
    }

    func flattenTree(node: BinaryTree) -> (leftMost: BinaryTree, right
        var leftMost = node
        if let left = node.left {
            var result = flattenTree(node: left)
            connectNodes(left: result.rightMost, right: node)
            leftMost = result.leftMost
        }

        var rightMost = node
        if let right = node.right {
            var result = flattenTree(node: right)
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```swift
class Program {
    // This is the class of the input root. Do not edit it.
    class BinaryTree {
        var value: Int
        var left: BinaryTree?
        var right: BinaryTree?

        init(value: Int) {
            self.value = value
            left = nil
            right = nil
        }
    }

    func flattenBinaryTree(root: BinaryTree) -> BinaryTree {
        // Write your code here.
        return root
    }
}

```

Our Tests

Custom Output Submit Code

```
        let expected = [3, 3]
        Prog.assertEqual(expected, actual)
    }
    runTest("Test Case 4") { () throws -> Void in
        let root = TestBinaryTree(value: 5).insert(value: 10)
        let leftMostNode = program.findLeastBinary(tree: root)
        let actual = leftRightTraversal(tree: leftMostNode)
        let expected = [5, 5, 5, 5]
        Prog.assertEqual(expected, actual)
    }
    runTest("Test Case 5") { () throws -> Void in
        let root = TestBinaryTree(value: 5).insert(value: 10).
        let leftMostNode = program.findLeastBinary(tree: root)
        let actual = leftRightTraversal(tree: leftMostNode)
        let expected = [5, 5, 5, 5, 5, 5]
```

Run or submit code when you're ready.