Our Solution(s) Run Code

Solution 1 Solution 2

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

public class Program {
  // O(n) time | O(d) space - where n is the number of nodes in the B
  // Tree and d is the depth (height) of the Binary Tree
  public static BinaryTree FlattenBinaryTree(BinaryTree root) {
    flattenTree(root);
    return getLeftMost(root);
  }

  public static BinaryTree[] flattenTree(BinaryTree node) {
    BinaryTree leftMost;
    BinaryTree rightMost;

    if (node.left == null) {
      leftMost = node;
    } else {
      BinaryTree[] leftAndRightMostNodes = flattenTree(node.left);
      connectNodes(leftAndRightMostNodes[1], node);
      leftMost = leftAndRightMostNodes[0];
    }

    if (node.right == null) {
      rightMost = node;
    } else {
      BinaryTree[] leftAndRightMostNodes = flattenTree(node.right);
      connectNodes(node, leftAndRightMostNodes[0]);
      rightMost = leftAndRightMostNodes[1];
    }

    return new BinaryTree[] {leftMost, rightMost};
  }

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```
public class Program {
  public static BinaryTree FlattenBinaryTree(BinaryTree root) {
    // Write your code here.
    return root;
  }

  // This is the class of the input root. Do not edit it.
  public class BinaryTree {
    public int value;
    public BinaryTree left = null;
    public BinaryTree right = null;

    public BinaryTree(int value) {
      this.value = value;
    }
  }
}

```

Our Tests

Custom Output Submit Code

Run or submit code when you're ready.