Our Solution(s) Run Code

Solution 1 Solution 2

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

#include <vector>
using namespace std;

class BinaryTree {
public:
  int value;
  BinaryTree *left = NULL;
  BinaryTree *right = NULL;

  BinaryTree(int value);
};

vector<BinaryTree *> getNodesInOrder(BinaryTree *tree,
                                     vector<BinaryTree *> *array);

// O(n) time | O(n) space - where n is the number of nodes in the Bina
BinaryTree *flattenBinaryTree(BinaryTree *root) {
  vector<BinaryTree *> inOrderNodes =
      getNodesInOrder(root, new vector<BinaryTree *>{});
  for (int i = 0; i < inOrderNodes.size() - 1; i++) {
    BinaryTree *leftNode = inOrderNodes[i];
    BinaryTree *rightNode = inOrderNodes[i + 1];
    leftNode->right = rightNode;
    rightNode->left = leftNode;
  }
  return inOrderNodes[0];
}

vector<BinaryTree *> getNodesInOrder(BinaryTree *tree,
                                     vector<BinaryTree *> *array) {
  if (tree != NULL) {
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```cpp
#include <vector>
using namespace std;

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

  BinaryTree(int value);
};

BinaryTree *flattenBinaryTree(BinaryTree *root) {
  // Write your code here.
  return root;
}

```

Our Tests

Custom Output Submit Code

```
class ProgramTest : public TestSuite {
public:
  void Run() {

    RunTest("Test Case 1", [](){
      BinaryTree *root = new BinaryTree(0);
      BinaryTree *leftMostNode = findLeftMostNode(root);
      vector<int> actual = leftRightSiblingTraversal(leftMostNode);
      vector<int> expected = {0, 0};
      assertIsEqual(expected == actual);
    });

    RunTest("Test Case 2", [](){
      BinaryTree *root = new BinaryTree(0);
      insertNRows(root, 20);
```

Run or submit code when you're ready.