Our Solution(s) Run Code

Solution 1 Solution 2

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

class Program {
  // O(n) time | O(d) space
  public static void invertBinaryTree(BinaryTree tree) {
    if (tree == null) {
      return;
    }
    swapLeftAndRight(tree);
    invertBinaryTree(tree.left);
    invertBinaryTree(tree.right);
  }

  private static void swapLeftAndRight(BinaryTree tree) {
    BinaryTree left = tree.left;
    tree.left = tree.right;
    tree.right = left;
  }

  static class BinaryTree {
    public int value;
    public BinaryTree left;
    public BinaryTree right;

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

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```java
class Program {
  public static void invertBinaryTree(BinaryTree tree) {
    // Write your code here.
  }

  static class BinaryTree {
    public int value;
    public BinaryTree left;
    public BinaryTree right;

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

```

Our Tests

Custom Output Submit Code

```
class Program{Test} {
  @Test
  public void Test1Case1() {
    TestBinaryTree test1 = new TestBinaryTree(1);
    Program.insertNBinaryTree(test1);
    InsertedBinaryTree insertedTest1 = new InsertedBinaryTree(1)
    Utils.assertTrue(isRepeatNBT(test1, insertedTest1));
  }

  @Test
  public void Test2Case1() {
    TestBinaryTree test2 = new TestBinaryTree(1);
    test2.insert(new int[] {2}, 0);
    Program.insertNBinaryTree(test2);
    InsertedBinaryTree insertedTest2 = new InsertedBinaryTree(1)
    insertedTest2.insert(new int[] {2}, 0);
    Utils.assertTrue(isRepeatNBT(test2, insertedTest2));
  }

  @Test
```

Run or submit code when you're ready.