COMP 587
Spring 2020

Automated Testing with Data Structures

1.) Consider the Java code below, which implements a binary tree:

```
public class Node {
  final int value;
  final Node left;
  final Node right;
  public Node(int value, Node left, Node right) {
    this.value = value; this.left = left; this.right = right;
  }

  // gets a random integer in the range [start, end)
  public static int randomInt(int start, int end) { ... }

  // null represents a leaf
  public static int getSum(final Node root) { ... }
}
```

1.a.) The following Java signature is intended to produce a random tree, suitable as an input to `getSum` above. Complete the code, which will generate nodes. You may use `randomInt`. For this part, don't worry about the behavior of `getSum`; only worry about generating `Node` values. As a hint, it's likely easier to write this in a recursive fashion.

```
// makes a tree which is, at most, maxDepth levels deep
public static Node makeTree(int maxDepth) {
```

1.b.) Describe how you could use `makeTree` to test `getSum`. If you need to make any modifications to `makeTree` for this testing, explain them, possibly with pseudocode.

1.c.) Now consider the following method, which is intended to get the depth of a given tree:

```
public static int getDepth(final Node root) { ... }
```

Describe how you could use makeTree to test `getDepth`. If you need to make any modifications to `makeTree` for this testing, explain them, possibly with pseudocode.