COSC-211: DATA STRUCTURES
HW7: BINARY SEARCH TREES

Due Thursday, April 5, 11:59pm

Reminder regarding intellectual responsibility: This is an individual assignment, and the work you submit should be your own. Do not look at anyone else's code, and do not show anyone your code (except for me and the course TAs).

1 The Assignment

In this assignment, you will write a binary search tree from scratch. Your tree should hold keys of type `String` (with no extra associated values). This is an unbalanced binary search tree—that is, it should obey the standard BST rules, but do not include any of the red-black tree balancing rules. You can use the `compareTo` method in the `String` class to determine which of two keys is larger.

Your tree should be in a class called `BinarySearchTree.java` (**exactly** like that, capitalization and all). In this file, you should include the following methods:

1. A method called `add` that takes a `String key` as input and inserts it into the appropriate position in your tree. If `key` already appears in the tree, the method should do nothing.

2. A method called `remove` that takes a `String key` as input and removes `key` from the tree. The method should return `key` if it successfully removed `key` from the tree, and it should return `null` if it did not remove `key` from the tree (i.e., if `key` was not in the tree).

3. A method called `lookup` that takes a `String key` as input and determines whether `key` appears in the tree. The method should return `true` if the tree contains `key` and `false` otherwise.

4. A method called `inOrderTraverse` that, when called on the root, prints all keys in the tree in increasing order. This method should have no input parameters and should not return anything.

Your methods must have **exactly the names, input parameters, and return types** specified above.

You might decide that you want to write additional methods, include fields, write additional classes, or any other number of things. This is all fine, provided that the four methods specified above do what they are supposed to do. Any methods or fields that you add to the `BinarySearchTree` class should be private.

You likely will want to write some code to test your binary search tree before you submit it. This code should be in a *separate Java file*; do *not* include test code in your `BinarySearchTree.java`. Some things to think about when you are testing your code: What happens when you try to call one of your methods on an empty tree? Does `add` work properly when you try to insert a key that

is already in the tree? Does removing the root work properly? This is not a comprehensive list of cases to check; you are responsible for making sure that you tree always works as described above.

2 Submit your work

Make sure you **test your code thoroughly** before submitting it. Code that does not compile will not receive credit.

Submit all of your Java files that I will need to use your tree (you do not need to submit extra code that you've written to test your tree) using either the submission web site or (from `remus/romulus`) the command line:

```
$ cssubmit *.java
```

This assignment is due on Thursday, April 5, 11:59pm.