COSC-211: Data Structures
HW7: Red-Black Tree Verification

Due Friday, November 18, 11:59 pm

(Note: I am setting this deadline on the 18th so that you can take a proper, homework-free break over Thanksgiving. There will not be a new assignment out until December 1, and you are welcome to take a penalty-free extension until then if you want. You may *not* use late days to submit the assignment any later than December 2.)

1 Testing *red-black tree* properties

Our goal, in this assignment, is to write a method that verifies that a red-black tree is *valid*. That is, the tree must fulfill the standard red-black tree properties:

1. Each node is colored *red* or *black*.
2. The root note is *black*.
3. Each null leaf is *black*.
4. A *red* node can have only *black* children.
5. At each node, the path to each null leaf must traverse an equal number of *black* nodes.

2 Getting started

Get started by creating yourself a directory for this project and grab some source code:

`https://bit.ly/cosc-211-f22-hw7`

Unzip the code, and you will see the following files:

- `RedBlackTree.java`

  A `RedBlackTree` object holds a pointer to a root node and implements some basic binary search tree operations. Note that this class **does not implement a full set of red-black tree operations**. It is a simple skeleton for building a binary search tree that *might* be a proper red-black tree.

- `RBNode.java`

  This class defines single a red-black tree node, providing child and parent pointers as well as a designation of node color.

- `RBTester.java`

  A special tester program for this particular assignment. This program reads and performs a sequence of `insert` operations, thus building the binary search tree described by the input sequence. The sequence (about which, more below) directs this program to build a tree with a particular set of values **and node colors**.

  Once the tree is constructed from the input, the program calls `isRBTree()` on the tree, determining whether the tree is valid and printing the results.

- A collection of `.txt` files

  These are a set of input sequence files that generate different (small) binary search trees with nodes labeled *red* or *black*. The ones beginning with the name `valid` create trees that should pass the `isRBTree()` test; the `invalid` ones should fail.

  Each of these is a text file that lists the order in which `insert()` is called on a tree; each line adds one value, and specifies whether the `RBNode` with that value should be marked *red* or *black*. You can create additional files of your own to create intentionally correct or incorrect trees, and then test whether your code detects these cases properly.

3 How to submit your work

Go to Gradescope for our course, where you can submit your work. It will be auto-tested, and you will see whether it *compiles* and *runs* successfully. As usual, if the run fails it won't tell you why; you need to go back and do more testing yourself. You may submit early and often!

Notice that **you should only submit** `RBTester.java`.

This assignment is due on Friday, November 18, 11:59 pm.

Again, note that this assignment is due on the 18th so that you can take a proper, homework-free break over Thanksgiving. There will not be a new assignment out until December 1, and you are welcome to take a penalty-free extension until then if you want. You may *not* use late days to submit the assignment any later than December 2.