Homework 6: Program Component

Due Monday, July 21, at 9 AM (GMT-7)

CSc 345 – Summer 2014

Instructor: Qiyam Tung (assignment by Kevin Coogan)

Instructions

1. This is an individual assignment. You must do your own work.
2. If you are having difficulty and need to ask a question you can:
   (a) Ask questions in class.
   (b) Stop by my office hours (or make an appointment).
   (c) Post a question on Piazza.
   (d) Post a private question on Piazza if the question is too specific.
3. Show all work. Incomplete solutions will **not** receive full credit
4. You may write your solutions by hand, or you may type them using any appropriate program such as Microsoft Word, OpenOffice Writer, LaTeX, etc...
   However, the final copy should be in PDF form and formatted so that it is legible.

1 Overview

This assignment is part of homework 6. It is worth 45 points.

You will implement a modified splay tree API. The modified splay tree will be similar to a normal splay tree, with one notable exception. A standard splay tree splays a node all the way to the root when it is searched, added, etc... The modified splay tree constructor will take an integer that will specify the maximum number of levels that a node will be splayed.

Hence, if a node is already closer to the root node than this limit, then it will be splayed to the root, as in a standard splay tree. Otherwise, it will move up the tree (using zig, zig-zag, and zig-zig operations) until its level has increased by the max limit. This may result in a zig operation in the modified splay tree to reach exactly max limit level changes, when in a standard splay tree a double rotation may have been required.

2 The API

To avoid confusion, an incomplete API file named ModSplayTree.java has been provided on the class web page. This file contains all of the public methods that you will need to implement for this assignment, including the constructor, the add method, the search method, and the traversal printing methods. (Note that the remove method does **not** need to be implemented as part of this assignment).

You should **not** add or remove any public methods to the file. The given public methods are the only ones that will be called to test the functionality of your program. You may (of course), implement any number of private methods to assist in the implementation. In fact, this is strongly encouraged.

You may also find it necessary to create other files to assist in your implementation. For example, a Node class to hold the tree nodes seems like a good idea, and may be implemented in another file if you wish. It will be assumed that all needed files are in the same directory.

3 Additional Requirements

To test your files, I will write my own driver in java that uses the ModSplayTree data structure. I will add or search for values, then print out the different traversals to make sure that the tree is correct. This, of course, means that if your traversals are not correct, I cannot tell if your tree is correct, so you will not receive credit.

Your program **must** handle duplicate key values. To make sure everyone's output is consistent, when adding a value to the tree that already exists, assume that it is greater than the previous value. That is, place it in the existing value's right sub-tree (before the splay, of course). Also, when searching for a value, you only need to splay the first instance of the value that is encountered.

Your hard copy will also be graded for documentation, and for design.

4 Program Input

Since we are implementing an API, there really are no input requirements. Inputs will come from the test driver's calls to the constructor, add, search, etc... methods. However, you should make sure that the API's public methods do all necessary bounds checking, and gracefully handle any problems that may come up. Remember, your program should never exit with an unhandled exception.

A sample test driver will be provided on the class web page, though it will be a stripped down version and will not test all cases. You are responsible for making sure the API is fully functional.

5 Output

Program output will be achieved through your implementation of the in, pre-, post-, and level order traversal printing methods. These methods should print all items in the tree on the same line, separated by a single space.