CSSE230 Exam 2 Practice (from Winter 2016-17) Name:_____

You may use any printed or written materials that you brought with you, code that you wrote yourself or with a partner before the exam, anything on the CSSE230 Moodle and web sites, and the Java API documentation (on your laptop or at oracle.com).

You may not otherwise search the internet or communicate with any person other than your instructor or a student assistant; no communication devices allowed. You may not communicate with anyone else about this exam until this afternoon.

You must actually get these problems working on your computer. All of the credit for each will be for code that actually works and for its efficiency. Use parameters and return values to pass information between methods (adding fields to the Node or BST class can cause you to lose many points for the problem). The parameters and return values can be of any type you like, including custom classes that you define.

To encourage elegant code where possible, we allocate a few points for code elegance (or lack thereof). Certainly, you will lose points if it appears that you just copy-and-pasted code that is overkill for the given problem. Also, you will lose most unit-test points if your code is hard-coded to work on the unit tests only.

Turn-in checklist:

1. _____ I did not give or receive help on this exam. Signature: _____

2. _____ I saved my work and committed my solution via SVN.

3. _____ I saw in Eclipse that my solution was committed.

4. _____ I am about to hand this test paper, with my name on it, to my instructor.

Problem	Possible	Score	Comments
1	17		
2	17		
3 unit tests	17		
3 efficiency	5		
Elegance	4		4=elegant, 3=OK, 1 or 2 = minor issues, 0=major issues, like redundant code or using Exceptions in expected situations.
Total	**60**		

Do all work in the BinarySearchTree class in the Exam2a repository.

Note: we use a BST of **Integer** rather than a BST of generic type T on this exam, to simplify the problems.

1. (17 points) Write **int countOneChildParents()** that returns the number of nodes having only 1 child. For instance, on buildTree1() below, it should return 3 (for parent nodes 70, 80, and 95.)

2. (17 points) Write **void pruneLeaves().** It should remove all leaves from the tree.

3. (17 points + 5 points efficiency) Write **Integer branchPoint(Integer x, Integer y)** that returns the value of the node at the "branch point" in binary search for x and y: that is, where a binary search for value x would last overlap with a binary search for value y. If both x and y are in the tree, this is equivalent to returning the "deepest common ancestor"—which can be x, or y, itself if one is an ancestor of the other. However, even if x and/or y is not in the tree, the branch-point should be well-defined based on the unsuccessful search paths… unless both searches end at the same NULL_NODE location, in which case you should return null.

   This will always be called on BinarySearchTrees. The efficiency points will be for writing code that runs in O(height) time, and does not perform redundant or unnecessary work. Efficiency will be checked manually by the instructor.

   (4 points) for elegance over all problems. Checked manually by the instructor.

   The following trees are used in test cases for problems 2 and 3, and may help you plan your algorithms.

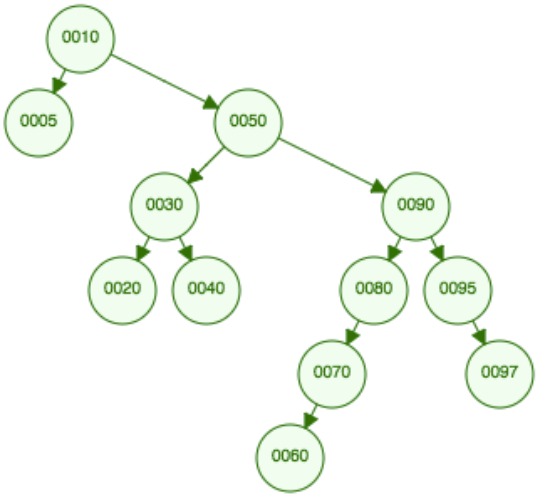

Figure 1. result of buildTree1()

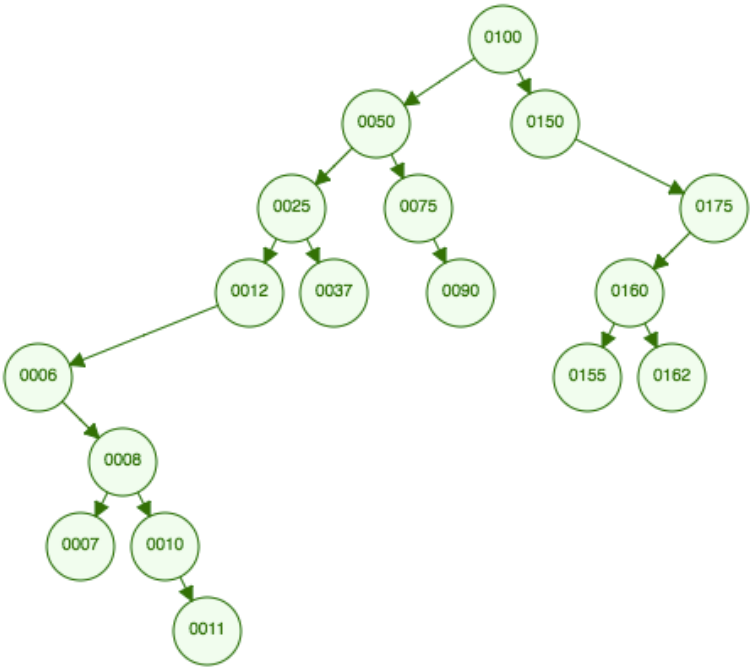

Figure 2. result of buildTree2()