CSE 326: Data Structures
Assignment #3
October 21, 2009
due: Thursday, October 29, at the beginning of quiz section

1. Do exercise 1 on page 251, but for ordinary binary search trees rather than AVL trees. That is, use Algorithms 6.8 and 6.9. You need only show the final tree for each of parts (a) and (b).

2. Explain exactly what is wrong with Algorithm 6.9 when the node N that contains the key K satisfies $\mathrm{LC}(\mathrm{RC}(N)) = \Lambda$. Fix the problem by replacing the final simultaneous assignment statement by statements that follow the suggestion of exercise 39 on page 214.

3. (a) Page 251, exercise 1a. Show the AVL tree after each insertion (including possible rotations) is completed.
   (b) Page 251, exercise 1b.
   (c) Show the result of deleting the key 538 from the resulting tree of problem 3b.

4. Page 251, exercise 2a. Show the AVL tree after each of the 10 insertions (including possible rotations) has completed.

5. (a) Number the nodes of the last tree in Figure 7.2 according to the order in which they are visited in an inorder traversal.
   (b) Using the numbers from part (5a) as the keys, show the result of doing an AVL-TreeDeleteMin on this AVL tree. Show the tree after each rotation. (For your information, it is not hard to see that executing a DeleteMin on these worst case trees causes $\Theta(\log n)$ rotations to occur. You might be interested in proving this to yourself.)

6. (a) Explain why it is much more convenient for AVLTreeDeleteMin to be written recursively than iteratively.
   (b) In executing AVLTreeDeleteMin, it is possible that there are rotations at a node v and some (distant) ancestor u of v, but no rotation at any of the nodes on the path between u and v. State exactly under what conditions this can happen.
   (c) Describe at what point you would invoke AVLTreeDeleteMin as a subroutine in the general AVL tree deletion algorithm of page 228, and how you would use the three values it returns.