Due: Tue Oct 18, in class.

INSTRUCTIONS:

• You will get some practice on AVL trees from doing this exercise, but unfortunately, the solutions cannot be published before the midterm.

• To get some solved answers on this topic, I suggest you look at the old homework and old midterms.

1. (1 Point) TRUE or FALSE: Recall that a rotation can be implemented with 6 pointer assignments. Suppose a binary search tree maintains successor and predecessor links (denoted $u.succ$ and $u.pred$ in the text). Now rotation requires 12 pointer assignments.

2. (4 Points) The text gave a conventional algorithm for successor of a node in a BST. Give the rotation-based version of the successor algorithm.

3. (3 Points) Give the in-order, pre-order and post-order listing of the nodes in the tree in Figure 15 (Lecture II).

4. (6 Points) Give a recursive routine called $CheckBST(u)$ which checks whether the binary tree $T_u$ rooted at a node $u$ is a binary search tree (BST). You must figure out the information to be returned by $CheckBST(u)$; this information should also tell you whether $T_u$ is BST or not. Assume that each non-nil node $u$ has the three fields, $u.key$, $u.left$, $u.right$.

5. (4 Points) Draw an AVL $T$ with minimum number of nodes such that the following is true: there is a node $x$ in $T$ such that if you delete this node, the AVL rebalancing will require two rebalancing acts. Note that a double-rotation counts as one, not two, rebalancing act. Draw $T$ and the node $x$.

6. (6 Points) Insert into an initially empty AVL tree the following sequence of keys: 1, 2, 3, ..., 14, 15.
   (a) Draw the trees at the end of each insertion as well as after each rotation or double-rotation. [View double-rotation as an indivisible operation].
   (b) Prove the following: if we continue in this manner, we will have a complete binary tree at the end of inserting key $2^n - 1$ for all $n \geq 1$.

7. (4 Points) Draw two AVL trees, with $n$ keys each: the two trees must have different heights. Make $n$ as small as you can.

8. (12 Points) Relaxed AVL Trees
   Let us define $AVL(2)$ balance condition to mean that at each node $u$ in the binary tree, $|balance(u)| \leq 2$.
   (a) Derive an upper bound on the height of a AVL(2) tree on $n$ nodes.
   (b) Give an insertion algorithm that preserves AVL(2) trees. Try to follow the original AVL insertion as much as possible; but point out differences from the original insertion.
   (c) Give the deletion algorithm for AVL(2) trees.

9. (0 Points) Let $T$ be the AVL tree in Figure 3(a) (page 11, Lect.II). This calls for hand-simulation of the insertion and deletion algorithms. Show intermediate trees after each rotation, not just the final tree.
   (a) Delete the key 10 from $T$.
   (b) Insert the key 2.5 into $T$. This question is independent of part (a).
      Re-do parts (a) and (b), but using the AVL tree in Figure 3(b) instead.

10. (0 Points) Describe what changes is needed in our binary search tree algorithms for the exogenous case.