EECS 311: Data Structures and Data Management **Homework 4**
Assigned: 10/17/08 **Due: 10/24/08**

1 Reading.

Read the supplemental splay tree reading posted on Blackboard.

2 Problems.

1. Show the trees that result from inserting keys 1, ..., 10, in order, into an initially empty AVL tree. Show the tree after each insert.

2. Show the trees that result from inserting keys 1, ..., 10, in order, into an initially empty splay tree. Show the tree after each insert.

3. Given a splay tree implementation of a dictionary, give an algorithm for *find-min* that returns the smallest key in the splay tree. You may assume that the standard splay tree operations, *create*, *insert*, *delete*, *find*, and *splay* have been already implemented. Prove that your find-min algorithm has amortized runtime of $O(\log n)$, i.e., that any sequence of m operations on an initially empty dictionary has worst-case runtime $O(m \log n)$ where n is an upper bound on the number of keys in the tree at any one time. (Hint: You should be able to use the in-class analysis of splay trees as a black box to prove the runtime. The proof is very short.)

4. Given a splay tree implementation of a dictionary, give an algorithm for *merge* that takes two dictionaries and merges them together. You may assume that the standard splay tree operations *create*, *insert*, *delete*, *find*, and *splay* have been already implemented. Prove that your merge algorithm has amortized runtime of $O(\log n)$, i.e., that starting with zero dictionaries, any sequence of m operations (including the creation of empty dictionaries) has worst case runtime of $T(m \log n)$ where n is an upper bound on the number of keys in any tree at any one time. (Hint: You should be able to use the in-class analysis of splay trees as a black box to prove the runtime. The proof is very short.)