**CSE 373 Winter 2015: Midterm Review List**
**One 8.5" × 11" Page (Double-Sided) of Notes Allowed**

1. Mathematical Foundations: Be able to give a proof by mathematical induction that a given function or procedure performs correctly based on induction with respect to some integer variable, say $n$ or *size* or *height*. The variable, in cases of our data structures, will be either the length of a list or array, the number of nodes in a tree, or the height of a binary tree.

2. Complexity

   - Given a function $f(n)$, be able to prove that it is $O(g(n))$ by using the formal definition and finding an appropriate $c$ and $n_0$.
   - Be able to perform an analysis of a given algorithm to determine the "number of statements executed" $T(n)$ by the algorithm for some given number of inputs $n$. Be able to convert this result to Big-O notation. Be able to analyze either iterative or recursive procedures. (You will not need to formally solve recurrence relations.)
   - Be able to compare the time complexities of various standard algorithms using Big-O notation.

3. Lists, Stacks, and Queues

   - Be familiar with the basic operations for lists, stacks, and queues; be able to use them as needed.
   - Be able to compare the algorithms for these operations with respect to sequential and linked implementations. Comparisons can be about what they do, the time complexity, and the required space.
   - Be able to write or analyze the complexity of recursive or nonrecursive procedures dealing with linear structures.

4. Trees

   - Be familiar with the abstract operations for binary search trees. Be able to use them as needed or to show what they do to a given tree.
   - Be able to write simple recursive or iterative functions that operate on general trees, plain binary trees, or binary search trees.
   - Be able to compute balance factors for the nodes of binary search trees.
   - Be able to show how the Insert operation works on an AVL tree, including the rebalancing operations for the 4 different cases.
   - Be able to explain the time complexity of any of the above algorithms.

5. Priority Queues as Heaps

   - Be able to show how insert and deleteMin work for binary heaps.
   - Be able to show how the buildHeap operation works, given some data in an array.
   - Be able to explain the time complexity of these algorithms

6. Union Find and Up-Trees

   - Be able to show how union and find work
     (a) for standard union where the second argument tree is hooked on to the root of the first
     (b) for union-by-size
     (c) for find with path-compression

7. General:

   - Be able to give short answer to questions about the structures and concepts we have covered.
   - Be able to write short code segments to do operations on the different structures we have studied