CSE 373 Final Review List
OPEN BOOK, OPEN NOTES

1. Complexity
   - Be able to analyze and compare the time complexities of various algorithms using Big-O notation.
   - Be able to answer questions on the class of polynomial-time algorithms, the class of NP-complete problems, and the concept of undecidability.
2. Lists, Stacks, and Queues
   - Be able to work with these structures, using abstract operations or implementing new operations as needed.
3. Recursion
   - Be able to trace how a given recursive procedure, function, or definition works on given input(s).
   - Be able to write a recursive procedure or function to accomplish some task, particularly involving the structures studied since the midterm: heaps, priority queues, and graphs.
4. Trees
   - Be able to write recursive or iterative functions that operate on general trees, plain binary trees, binary search trees, or B-trees with a given node structure.
   - Be able to answer questions on balancing techniques.
5. Hashing
   - Be able to show how open addressing works with various collision-handling schemes (linear probing, quadratic probing, double hashing, rehashing or some given scheme) on given data.
   - Be able to show how extendable hashing works on given data.
   - Be able to use hashing as a utility in the solution of application problems.
   - Be able to analyze the complexity of given hashing schemes or algorithms that use them.
6. Priority Queues
   - Be able to apply the basic operations Insert and DeleteMin to a binary heap.
   - Be able to build a heap (either min or max) using the BuildHeap approach.
   - Be able to write or analyze functions that work with binary heaps.
   - Be able to show how Heapsort works on a small example.
7. Disjoint Sets
   - Be able to use the union-find data structure in problems that deal with disjoint sets.

- Be able to answer questions about the various union and find variations and their complexity.

8. Graphs and Digraphs
    - Be able to write functions that work with any of the the variations: directed graphs, undirected graphs, weighted and unweighted graphs.
    - Be able to use the two different representations we covered: adjacency matrices and adjacency lists.
    - Be able to show how the following algorithms work on given data:
        - breadth-first and depth-first search
        - topological sort
        - unweighted shortest path
        - Dijkstra's algorithm for weighted shortest path
        - Kruskal's algorithm for finding the minimal spanning tree of a weighted graph
        - the backtracking tree search algorithm for subgraph isomorphism
        - the branch-and-bound search for finding least-error mappings