MA/CSSE 473 Fall 2014 midterm exam specification (Last updated Sept 25, 2014)

Resources allowed:

1. Calculator (not a smart phone, iPod, etc. that happens to have a calculator.
    a. The only necessary operations are +, *, -, /, exponentiation, and you probably won't need those.
2. One page of notes (handwritten on one side).
3. No other electronic devices, including phones, MP3 players, anything with headphones or earbuds.

Material Covered:

- **Background material from 230:** Algorithms and analysis for implementing stacks, queues, linked lists, sequential lists, binary trees, binary search trees, binary heaps, dictionaries. Recursion and mathematical induction. Writing and solving simple recurrence relations, analysis of nested loops as in Weiss chapter 5 and its exercises. Fibonacci numbers. Sequential and binary search. Well-known sorting methods: Insertion, selection, merge, quick, heap. Binary tree traversals: preorder, inorder, postorder, level order. Formal definitions of O(N), Θ(N), etc.
- **HW 0 - HW 5** (including the "not to turn in" problems)
- **Assigned readings** (Chapters 1-4 from the textbook as well as documents posted on Moodle that are listed on the Schedule page for Sessions 1-12), and material from PowerPoint slides for sessions 1-14.
  More details below on specific algorithms you should know.

Question types:

T-F-IDK. Below you will find several statements. A statement is true (T) if it is always true. It is false (F) if there is at least one counterexample (sometimes false). You may also choose IDK to indicate that you do not know the answer. *There is some value (to me and you) in knowing what you don't know.*

Point values: Correct answer: 5, incorrect answer: 0, IDK: 3, blank: 0. If you are totally guessing an answer, you are probably better off pointwise if you choose IDK. If you have a feeling that one "real" answer is correct, you are probably better off guessing that answer.

Do you know the details of this algorithm? Some questions will be checking to see whether you know and understand the details of specific algorithms. Here are examples of algorithms that you should be able to explain and discuss the results of what happens when they are implemented with specific input data.

Examples (there are more examples in the background material section above)**:** **(the first 6 bullets are form Dasgupta)**

- Addition, multiplication, exponentiation of integers (bit by bit, digit by digit), modular arithmetic.
- Euclid's algorithm for finding GCD
- Extended Euclid for finding modular inverse, and the multiples that lead to GCD.
- Fermat's little theorem. Uses and limitations when doing primality testing.
- Carmichael numbers, probabilistic primality testing.
- RSA public-key cryptography
- Sieve of Eratosthenes (L 1.1)
- Towers of Hanoi (L 2.4)
- Selection Sort, Bubble Sort, and their analysis (L 3.1)
- Sequential Search (L 3.2)
- Brute force String Match (L 3.2)
- Brute Force Convex Hull (L 3.3)
- Brute Force Closest Pair (L 3.3)
- DFS and BFS in a graph (L 3.5)
- Fast exponentiation (L 4.1)
- Insertion Sort (L 4.1)

- Topological Sort (L 4.2)
- Johnson-Trotter algorithm for generating permutations (L 4.3)
- Lexicographic order permutation generation (L 4.3), from permutation to number and back (my slides)
- Subset generation and binary reflected Gray code (L 4.3)
- Binary Search (L 4.4)
- Fake Coin Problem (L 4.4)
- Russian Peasant Multiplication (L 4.4)
- Josephas Problem (L 4.4)
- QuickSelect (L 4.5)
- Interpolation Search (L 4.5)
- Binary tree search and insertion (L 4.5)

Concepts, definitions, etc

- Graph representations (adj matrix, adj lists) (L 1.4)
- Graph definitions: edge, directed, head tail, digraph, loop, complete, dense, sparse, path, simple path, cycle, connected component (L 1.4)
- Efficiency: best case, worst, average, amortized (L 2.1)
- Asymptotic notation (O, Ω, Θ), including formal definition of big-O, limits and asymptotics (L 2.2)
- Nested loops and summation (L 2.3, W Chapter 5)
- Traveling Salesmen problem, Knapsack Problem, Assignment problem (L 3.4)
- Master Theorem for divide-and-conquer algorithms. No need to memorize; I will give you the formulasn(L 5.1)

Can you come up with a reasonably efficient algorithm to … ? Many of the homework problems (especially the puzzle problems) have been this type of problem.

Analyze the running time of this algorithm