CS/Math 240: Introduction to Discrete Mathematics	7/10/2007
Exam 1 Handout	
Instructor: Jeff Kinne	TA: Mike Kowalczyk

The first exam will take place during the regular class time on Monday, July 16. It will be closed book, closed notes, and no calculators allowed. You can prepare a 4x6 inch index card as a "cheat sheet" to use during the exam. You can write whatever you want onto it, or you can print off of a computer and paste onto the 4x6 card. The instructor will bring extra paper if you want scrap paper to write on.

Material on the Exam

The exam will cover material that was covered in lecture from the first day of class through Monday, July 9. This material can be broken up into a few major parts.

Mathematical Background

We introduced a number of different mathematical objects that we use throughout the course. For each, you should know the definition of what these objects are. We have also listed for each type of object a number of terms or properties that are related to that type of object. For properties of the objects, you should be able to determine (and prove) whether an example object has the property or not - for example, prove whether a function is 1-1 or not.

- **Logical Operators:** truth table, know meaning of logical operators - (and, or, not, implication, $\exists$ and $\forall$),
- **Sets:** subset, union, intersection, set difference, complement, De Morgan's laws, distributive law (of union over intersection and vice versa), cardinality, countable, uncountable, Cartesian product of sets, Venn diagram reasoning, cardinality, power set, inclusion-exclusion rule for 2 and 3 sets,
- **Functions:** one-to-one, onto, one-to-one correspondence, domain, range, co-domain
- **Graphs:** directed, undirected, connected, bipartite, path, graph representation (adjacency matrix, adjacency list), degree of a vertex, weighted graph,
- **Relations:** transitive, symmetric, reflexive, equivalence relation, anti-symmetric, partially ordered set,
- **Algorithm:** correctness, running time,

Algorithms

We have spent about half of the lectures studying algorithms. These have been mostly graph algorithms, but there have been a few others as well. For each of these algorithms, you should understand the algorithm enough to be able to trace through it on an example, and you should be able to prove the correctness of the algorithm if given the code for it (using a loop invariant and/or induction).

- **List Search:** searching an unordered list for some element,
- k^{th} **Smallest Element:** from the second homework assignment,
- **Binary Search:** searching an ordered list for an element by dividing the list in half,
- **Merge Sort:** sorting a list by dividing in half, sorting each half, then merging the lists,
- **Breadth First Search:** can be used to test connectivity, can also be used to get a spanning tree,
- **Dijkstra's Algorithm:** finding shortest path in weighted graphs that do not have negative weights,
- **Topological Sort:** algorithm given in class to topologically sort any finite partial order,
- **Prim's Algorithm:** finding a minimum spanning tree in a weighted connected graph.

We saw a few techniques for designing algorithms. Dijkstra's and Prim's algorithms are *greedy* - they make decisions that seem good when looking at a local view of the graph, and those decisions end up producing a total solution that is good. Merge sort is a *divide and conquer* algorithm - it splits the job to be done into smaller pieces, solves those pieces, then gets the final solution by combining them. We gave a *recursive* definition of binary search - where we reduce solving the problem on inputs of length m to the problem of solving the problem of inputs of length smaller than m. I only state this for your information - you won't be tested on your knowledge of "greedy algorithms", for example.

Proof Techniques and Tools

We introduced a few proof techniques that we have used in the first half of the course. You should be able to use these on the exam.

- **Proof by Contradiction:** If you are trying to prove that statement P implies statement Q, you assume that both P and the negation of Q hold. If you can come to a contradiction, it must be that P implies Q. This is similar to trying to prove the contrapositive of a theorem.
- **Proof by Induction:** For some statement $\mathrm{P}(n)$ such as $\mathrm{P}(n) =$ "merge-sort is correct on all lists of size at most n", show that $\mathrm{P}(n)$ is true for all integers n at least some fixed value k by showing that: 1) $\mathrm{P}(k)$ is true, 2) if we know that $\mathrm{P}(n)$ is true for any $n \geq k$, then $\mathrm{P}(n+1)$ must also be true.
- **Propositional Logic:** You should be able to translate an English sentence into logical form, and vice versa. Given a list of valid rules of inference (you don't have to remember these), you should be able to determine if a simple line proof is correct. You should be able to give a truth table proof that two propositional statements are equivalent (for example p and $\neg\neg p$).
- **Big-O, Big-Ω, Big-Θ:** You should be able to give these estimates for functions by applying the definition of them (either the definition from class or from the second homework).

- **Solving Recurrence Relations:** We looked at recurrence relations as a way to define the running time of recursive algorithms. Recurrence relations can also be used to define functions recursively (such as the Fibonacci sequence). You should be able to use both the Master Theorem if given to you (you don't need to remember it) or a recursion tree to solve a recurrence relation.

Miscellaneous

There are certain miscellaneous facts that I will assume you know. If you don't know these, I suggest you write them on your cheat sheet.

- **Formulas for Arithmetic and Geometric Sums**
- **Telescoping Sums:** be able to look at a telescoping sum and cancel terms to get the final answer if all but a few terms cancel.
- **Rules of Exponents**
- **Rules of Logarithms**
- **Basic Sets:** Integers, Naturals (includes 0, but not negative numbers), Reals, Rational numbers, Irrational numbers.

Study Helps

There are a number of sources that you can look to in studying for the exam.

- **Quizzes:** There are model solutions in the question feedback for each daily quiz.
- **Homeworks:** You have the model solutions from the first two homeworks, and you will get the model solution for the third homework assignment at the end of class Thursday. The exam questions won't necessarily be that difficult, but they're still a good reference.
- **Practice Exercises:** The practice exercises listed on the course website for each lecture would be excellent practice. They are all odd problems, so the answers are in the back of the book.
- **Sample Exam:** There is a sample exam on the course website that is also handed out in class. These problems will give you an idea of the difficulty of the questions. I suggest trying them out on your own, and then going to the exam review session to see how you did.
- **Exam Review Session:** Mike will hold an exam review session on Thursday, July 12 from 12:00-1:30pm in 1325 CS. He will answer any questions you have, including going through the sample exam.
- **Suggested Readings:** The suggested readings for each day covers all the material we covered in class, in addition to some examples that we didn't cover in class. I point out that sometimes the suggested reading was modified after a lecture if we covered slightly different material in class than originally planned.