Final Exam
What you need to know

CMSC 203

- *This review sheet will outline the basics of what you should know for the exam to do well. Familiarity with these topics is a **minimum** for performing well, but questions may appear on the exam that are not on this list. Be sure to review all the material we have covered in class.*
- *To practice, verify you can successfully complete the homework problems and odd numbered problems in the book without assistance.*
- *A cheatsheet will be provided for you, and calculators **will** be allowed and highly recommended.*

1 Propositional ∧ Predicate Logic

- **Verify** if an English statement is a proposition, and whether it is a tautology or contradiction
- **Convert** between English and propositional / predicate logic
- Given a propositional statement, produce a **truth table**
- Determine if two logical statements are **equivalent** by using rules of inference

2 Proofs

- Recognize and know the various **rules of inference** and mathematical axioms (such as even numbers)
- Execute a **proof** method correctly (Direct Proof, Contrapositive, Contradiction, Cases, Exhaustive, Existence, Uniqueness)

3 Sets

- **Convert** between roster and set builder notation
- Prove **equality** of two sets
- Understand **set operators**, and be able to convert statements to and from a **Venn Diagram**

4 Functions

- Understand **Domain**, **Codomain**, **Preimages**, **Images**, and **Range** of a function
- Know whether a function is **one-to-one**, **onto**, or **one-to-one correspondence**

5 Algorithms and Complexity

- Prove $f(x)$ is or is not $O(g(x))$ using a **proof by existence**
- Use **simplifying assumptions** in asymptotic analysis to find big-O
- Analyze the **asymptotic runtime** of a procedure
- **Compare and rank** procedures and runtime of functions using big-O
- **Develop** an algorithm to solve a problem, and give the runtime

6 Induction and Recursion

- Prove statements using **mathematical induction** and **strong induction**
- Develop a **recursive algorithm** to solve a problem, and prove it is correct

7 Counting

- How to appropriately use the **product rule**, **sum rule**, **subtraction rule**, and **division rule** for a real-world problem
- Using the **Pigeonhole Principle** to find the number minimum number of items that must be in at least one box (solving for both N and k)
- Solve real-world **permutation** and **combination** problems with and without repetition
- Determine the **binomial expansion** of some $(x+y)^n$, and finding the arbitrary **coefficient** for some binomial problem using mathematics and Pascal's Triangle

8 Probability

- Find the **probability** and **conditional probability** of real-world problems, including **non-uniform** distributions
- Show whether two events are **independent** or **dependent**
- Use **Baye's Theorem** to solve for conditional probabilities
- Understand **random variables** and how to find the distribution of random variables
- Find the **expected value** of real-world problems, such as monetary return or number of actions

9 Relations

- Know the **properties** of relations (Reflexive, Symmetric, Antisymmetric, Asymmetric, Transitive) and how to determine if they apply
- Determine what elements are or are not in a relation
- **Convert** relation representations between sets of order pairs, matrices, and digraphs
- How to represent multiple domains as an **n-ary** relationship
- Find valid **primary** or **composite** keys for an n-ary relationship
- Apply **operators** to an n-ary relationship (Selection, Projection, Join)
- Understand how operations translate to **SQL statements**

10 Graphs

- Know whether a graph is **directed, multigraph, reflexive, or weighted**
- **Create** a graph to represent a real-world problem
- **Represent** a graph as an image, an adjacency list, or an adjacency matrix
- Know the **special graphs** and how to represent them
- Determine if a graph is or is not **isomorphic**
- Determine the **connectedness** of a graph, and find potential **cuts**
- Solve a shortest-path problem using **Dijkstra's Algorithm**