EECS 268 Review Materials for Exam 2 Spring, 2009

Administrative

- When: 9:30 - 10:45 am, Tuesday, April 7, 2009.
- Where: 1136 Learned.

Read carefully all of the following items....

- Bring your KUID. You will not be allowed to take the exam without your KUID.
- The exam will be closed book and closed notes.
- No calculators, cell phones, head phones, or electronic devices of any sort will be allowed. No such devices should be out in the open.
- Once you start the exam, you will not be excused from the room for any reason unless you turn in your exam. Once it is turned in, you cannot come back and continue working on it.

Exam Coverage

- Chapters 5-9 of Carrano.
- Material covered in lectures notes 5-9 and labs.

Topic Review

Chapter 5

- Understand the basic structure of recursive definition and its implementation using recursive programming technique.
- Know the general characteristics of the major recursive algorithmic design strategies we have studied including:
  - Divide and Conquer.
  - Backtracking.
- Understand how the Queens problem works using backtracking.
- Understand grammar and be able to formally describe what a grammar is for simple languages.
- Given a grammar for a language, be able to:
  - Identify the language generated by the given grammar.
  - Generate by hand strings in the language according to some criteria (e.g., "generate a seven-character string in this language").
  - Implement a language recognizer (a recursive function that can determine whether a candidate string is in the language).
- Understand infix, prefix, and postfix algebraic expressions. Given an expression specified in one, be able to generate by hand the equivalent expression in one of the other forms. Be able to evaluate an expression in any of these forms.
- Be able to process and implement list operations recursively.
- Be able to read recursive code and predict what will happen when it executes.

Chapter 6

- Understand the Stack ADT at both the implementation and user levels:
    - Understand the major implementations (array based, linked list based, ADT List based) of stack class.
    - Given one of the three implementation approaches, be able to write the code implementing any of the Stack methods (including constructors and the destructor).
    - Given a problem for which a Stack is required, be able to use the public Stack ADT interface *exactly* as studied in class and presented in the book to solve the problem. Such a problem may be one from the book, one from the exercises, or something different.
    - Understand and be able to use stack to convert an infix expression to prefix and postfix expression. Be able to evaluate an expression in infix, prefix, and postfix forms using stack.
- Be able to read stack-based code and predict what will happen when it executes.

Chapter 7

- Understand the Queue ADT at both the implementation and user levels:
    - Understand the major implementations (array based, linked list based, ADT List based) of queue class.
    - Given one of the three implementation approaches, be able to write the code implementing any of the Queue methods (including constructors and the destructor).
    - Given a problem for which a Queue is required, be able to use the public Queue ADT interface *exactly* as studied in class and presented in the book to solve the problem. Such a problem may be one from the book, one from the exercises, or something different.
- Be able to write code which uses queue or a combination of stack and queue.
- Be able to read queue-based code (and/or code which uses both stack and queue) and predict what will happen when the code executes.

Chapter 8

- Understand different types of inheritance and what they are used for.
- Understand the visibility and accessibility rules for different types of inheritance.
- Be able to write code using inheritance.
- Be able to execute and predict what will happen when code with inheritance executes.
- Understand and know the difference between function overloading and function redefinition. Be able to redefine a function inherited from a base class.
- Understand the basic properties of virtual function, pure virtual function, abstract class, static and dynamic type of variables, and late binding.
- Understand virtual methods and runtime bindings. Know how to predict what methods will be called in given situations. Be able to write class definitions with virtual methods.

Chapter 9: Algorithmic Complexity and Sorting

- Understand the concept of algorithmic efficiency and their applications.
- Understand and be able to define closed-form expression, big-O and big-Θ relations.
- Understand the basic concepts of best case, worst case, and average case complexities of algorithms.
- Understand the performance and asymptotic complexity of the sorting algorithms we studied and know the conditions under which each algorithm achieves its best, average, and worst case performances. We may ask you to supply the complexity functions for some sorting algorithms and/or also provide an input to a sorting algorithm so that the best/worst case complexity can be achieved.
- Understand and be able to describe how each of the sorting algorithms works. We may give you a series of "snapshots" of an array while it is being sorted and ask you what algorithm is being used to sort the array.

Remark: For problems in which we give you code, be able not only to state what happens when the code executes, but also be able to locate bugs in the code.

Hints

- Be sure to read and study the Key Concepts, Summary, Self-Test Exercises, and Exercises throughout the chapter. (The Programming Problems are usually less helpful when studying for exams.)
- Review the class lecture notes and lab assignments.
- Read and be sure you understand Carrano's code, especially his implementations of ADT methods.