# EECS 268 Review Materials for Exam 1 Spring, 2009

## Administrative

- When: 9:30 - 10:45 am, Thursday, February 26, 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 1-5 of Carrano.
- Material covered in lectures notes 1-5 and labs.

## Topic Review

***Chapter 1***

- The Software Life Cycle model (What is it? What happens in the various steps?)
- Understand the importance of the two basic Enduring Principles: Abstractions and Algorithms.
- Designing modular solutions (What is abstraction? What is its role in modular design? What are the types of abstraction? What do we mean by information hiding, and why is it important?)
- Know the difference between a data structure and an abstract data type.
- Understand Top-Down and Object-Oriented Designs. In addition, know how the two work together.
- Understand the importance of testing and debugging.
- Be able to read UML and know how to translate a UML description into a C++ class definition.
- Understand the "Seven Key Programming Issues" on page 34.

***Chapter 2***

- Understand the four characteristics of recursive programs.
- Be able to write simple recursive functions according to a given specification.
- Be able to trace through recursive code using recursion trees and box diagrams.
- Understand different performance issues when designing iterative and recursive programs.
- When tracing a recursive function, be able to tell us things like "What will be output?" or "Will a given recursive call terminate in finite time? Why or why not?"

- Be able to construct simple recursive functions based on the design technique of divide-and-conquer.

***Chapter 3***

- Be able to define Abstract Data Type and explain its role in information hiding in software systems.
- Understand the basic approach in the design and development of ADT.
- Know the ADT: List and its array implementation. Understand different performance issues when implementing an ADT using arrays.
- Be able to implement standard list operations like insertion, removal, retrieval, and simple traversal based on array implementation.
- Be sure you understand what each ADT list operation is to do. For example, you may see a question like "Recall the ADT List class definition shown below. Give an implementation of method xxx, assuming an implementation based on a singly linked list data structure." We will simply assume that you know everything about what the xxx method is supposed to do. We will not explain that to you during the exam!
- Given the interface to a C++ class implementing, be able to: Use the public interface to accomplish some task implement methods of the class (This includes constructors, the destructor, as well as general instance methods.)
- Be able to implement basic list operations such as find, insert and delete operations.
- Understand the role of namespaces. Know how to create them and use objects which are defined within them. Be able to access the objects, both with a using statement as well as without one. Be able to predict the output of code which uses namespaces.
- Know how to define, throw, and catch exceptions. Be able to predict how code using exceptions will execute. For example, be able to predict the output of code which utilizes some combination of throwing and catching of exceptions. Be able to realize when an exception that is thrown will not be caught, and know the result when this happens.

***Chapter 4***

- Be able to define, initialize and use pointers.
- Understand the difference between static and dynamic variables.
- Know how to allocate and de-allocate dynamic variables.
- Understand and be able to identify and avoid the problems of dangling pointers and memory leaks.
- Know how to allocate and de-allocate dynamic arrays.
- Understand the differences among pointers, arrays, and dynamic arrays.
- Be able to use dynamic variables and dynamic arrays in applications.
- Know the ADT: List and its pointer implementation. Understand different performance issues when implementing an ADT using pointers.
- Be able to implement standard linked list operations like insertion, removal, retrieval, and simple traversal using pointer implementation.
- Be sure you understand what each ADT list operation is to do. For example, you may see a question like "Recall the ADT List class definition shown below. Give

an implementation of method *xxx*, assuming an implementation based on a singly linked list data structure." We will simply assume that you know everything about what the *xxx* method is supposed to do. We will *not* explain that to you during the exam!

- Know when and how to use `delete` in the implementation of list methods based on linked lists.

**Chapter 5**

- Understand and be able to use recursive programming technique to implementing various list operations.
- Know the general characteristics of divide-and-conquer and backtracking algorithms and be able to develop recursive algorithms/functions using these design methods.
- Understand how the Queens problem works and its solution using backtracking.
- Be able to use and demonstrate backtracking algorithms using backtrack search tree.
- Be able to read recursive code and predict what will happen when it executes.

## 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.