CS 215 - Fundamentals of Programming II
Spring 2010 - Final Exam Review Sheet

Notes and Reminders:

- Project 7 is due on Tuesday, April 27 at 4:30pm. Late projects will be accepted until Friday, April 30 at 4:30pm (normal 3 days late).
- Homework 10 is due on Monday, April 26 at the beginning of class. NO LATE SUBMISSIONS will be accepted.
- Monday, April 26, has been set aside as a review for the exam. We will go over the homework exercises since the midterm exam, and answer any questions you have about the material.

The Final Exam will be on Monday, May 3, at 2:45pm-4:45pm in KC-267. You may bring one 8.5inx11in size sheet of paper with notes on one side to the exam. You may print out the sheet, but it must be in a 9-point font or larger. E.g., please do not photoreduce or print 4 pages on a side. If you handwrite your notes, they may be as small as you like. You may handwrite notes in the margins of a printout.

The exam will be cumulative and comprehensive. You are expected to be able to read and write code or analyses and designs using concepts from the entire course. These concepts may be presented singly or in combination. Emphasis will on the material since the midterm exam in Chapters 6-10 (except 5.6, 7.5, 8.5, 8.6, 9.7, and 10.8), 11.1-11.3, 14.1-14.2, and 15.1 (QuickSort and Comparison of Sorting Algorithms only), and covered in lectures and assignments made through Friday, April 23. This material will comprise about two-thirds of the exam. You are not responsible for the material on implementing iterators. The exam will consist of questions similar to the homework problems, programming projects, and exercises in the textbook.

The following is a list of topics that will be emphasized, but it is in no way to be construed as an exclusive list. Consult the midterm review sheet for other topics.

1. Lists and iterators - declaration and use
2. Stacks and queues - declaration and use
3. Comparison of the STL containers studied
4. Linked lists - design and implementation of singly-linked lists, use as an implementation technique for other ADTs.
5. Construction and use of doubly-linked lists.
6. Construction and use of binary trees: linked implementation (as for code trees and binary search trees), and array implementation (as for heaps); iterative and recursive scan-based algorithms; binary search tree as an ADT.
7. Time complexity of data structure operations
8. Implementation and comparison of complexity of sorting algorithms: selection sort, insertion sort, bubble sort, exchange sort, heap sort, and quick sort.