CS 215 - Fundamentals of Programming II
Spring 2008 - Final Exam Review Sheet

Notes and Reminders:

- Homework 9 is due on Friday, April 25 at 4:30pm. NO LATE SUBMISSIONS will be accepted.
- Project 7 is due on Monday, April 28 at 4:30pm. Late projects will be accepted until Thursday, May 1 at 4:30pm (normal 3 days late).
- Monday, April 28, has been set aside as a review for the exam. We will go over Homeworks 7, 8, & 9, and answer any questions you have about the material.

The Final Exam will be on Friday, May 2 at 12:30pm-2:30pm 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 Exam 2 in Chapters 9 (except 9.7), 10.1-10.7, 14.1-14.2, and 15.1, and covered in lectures and assignments made through Friday, April 25. This material will comprise about one-half to 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 previous review sheets for other topics.

1. Linked lists - design and implementation, use as an implementation technique for other ADTs.
2. Construction and use of doubly-linked lists
3. 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.
4. Implementation and comparison of complexity of sorting algorithms: selection sort, insertion sort, bubble sort, exchange sort, heap sort, quick sort, and merge sort.