CSc 227 — Program Design and Development
Spring 2014 (McCann)

http://www.cs.arizona.edu/classes/cs227/spring14/

Final Exam Topic List (Additions Only!)

Purpose: You've received topic lists for each of the midterm exams. The final exam is comprehensive; those prior topics are fair game for the final, in addition to the topics listed below, which you have yet to be tested on. Merging the topic lists should give you a really good idea of what you'll have to know to do well on the final exam.

Your goal is to learn both the "how" and the "why" of all of these (and of the prior) topics. For example, you need to know what a circular linked list is and how it works, and why a programmer would want to use such a list representation instead of the other list representation options.

As always: Please note that this is not meant to be an exhaustive list of exam topics; rather, it's meant to hit the highlights and ensure that you don't overlook a critical topic.

1. Linked Lists
   (a) Pros and cons as compared with arrays
   (b) Implementations of common list operations
   (c) Using singly linked lists as representations of stacks and queues
   (d) Linked List Variations:
      i. Adding a tail reference
      ii. 'Dummy' nodes ahead of and behind the nodes holding actual data
      iii. Circularly-linked lists (the first node 'follows' the last node)
      iv. Doubly-linked lists (each node references both successor and predecessor nodes)
2. Recursion
   (a) The "what's simpler than . . . ?" approach to the expression of a problem's solution in terms of simpler case(s) of itself.
   (b) Base case(s) and general case(s)
   (c) Tracing recursion (using stacks of activation frames)
   (d) "Store-front" and "factory" two-part recursive solutions
3. Hierarchical Data Structures (a.k.a. Trees)
   (a) Tree terminology (root, parent, child, internal node, leaf node, siblings, degree, level, height)
   (b) General trees
   (c) Binary trees
      i. Expression trees
      ii. Tree traversals (preorder, inorder, postorder)
      iii. Binary search trees (BSTs) – searching, inserting, deleting
4. Algorithm Analysis
   (a) Best, worst, and average case performance of algorithms
   (b) Standard comparison functions
   (c) Big-O notation
5. Sorting
   (a) Simple sorts (Insertion sort, Selection sort)
   (b) Complex sorts (Treesort, `Collections.sort()`, and, if we have time, Quicksort)