CSE 143 Final Exam Topic List – Mar. 9, 2005

Here is a brief checklist of the additional topics that have been covered since the second midterm and might appear on the final exam. You remain responsible for everything earlier in the course. The final will be comprehensive, although it will tend to emphasize topics covered since the second midterm.

- Anything related to any of the projects, particularly project 3
- Efficiency and basic complexity theory
  - Problem size; notion of an abstract "step" or unit of work
  - Measuring work (time particularly) as a function of the problem size
  - Asymptotic complexity – why it matters
  - $O(\,)$ notation – understand the definition and how to show $f(n)$ is $O(g(n))$
  - Arithmetic with $O(\,)$ notation
  - Comparing implementations using $O(\,)$ (lists, trees, hash tables, etc.)
- Implementation of lists using linked lists
  - Tradeoffs vs array-based implementations
- Recursion
  - Base cases vs recursive cases; making progress towards termination
  - Use of helper functions to kick off the recursion (i.e., implementing `contains()` by calling `contains(lo,hi)` which does the real work)
  - Be able to calculate time of a recursive algorithm expressed in $O(\,)$ notation.
- Linear search and (recursive) binary search
- Sorting, particularly quicksort* – a recursive divide-and-conquer algorithm
  - Hint: know the ideas/pictures behind this, don't memorize code
  - Understand why simple sorts like insertion sort run in $O(n^2)$ time
  - Be able to explain why quicksort runs in $O(n \log n)$ time
    - How do you pick good pivot values to guarantee this for Quicksort?
- Trees, particularly binary trees
  - Know the vocabulary: nodes, edges, root, leaves, etc.
  - Tree traversals: preorder, postorder, and inorder
  - Know how to design/write recursive algorithms to process trees
- Binary search trees
  - Basic properties: when is a binary tree a BST?
  - Recursive algorithms to search for and add items to a binary search tree (what does the tree look like if the following values are inserted in this order….?)
  - Cost of operations on a binary search tree, both expected and worst-case, and what needs to be true to avoid worst-case behavior (balanced trees)
- Hashing and implementing simple sets with hash maps*
  - General idea behind hashing, hash functions, buckets
  - Cost of operations – expected, worst, and what affects the running time
  - The purpose of Java's `hashCode()` method specified in class `Object`, and the relationship between `hashCode()` and `equals()`

*How much you are responsible for about quicksort and hashing will depend on how extensively these are covered in the last week of class. This part of the list eill be revised if needed.