CSC 112 Final Exam: Study Guide

The final exam will be designed to be about 1.5 times the length of the regular tests, but you can work the whole 3 hours on the exam if you need or want to. Expect 9 or 10 questions. Reviewing the earlier tests and homeworks is probably one of your best ways to study – understand what knowledge the questions were asking you to understand (not just the details of the questions).

The following skills (roughly grouped by topic) obtained in the course may be tested on the final exam:

From Test 1:
- Many syntactical ideas (use of loops, use of arrays, dynamic memory allocation new and delete) will not be tested by themselves, but may pop up in other problems.
- An understanding of the function call stack, the heap, and variable placement in the stack and heap, including memory errors (memory leaks and dangling pointers)
- Understanding the workings of the array container, as well as being able to reason about appropriate types of arrays to employ in solving a problem.

From Test 2:
- An understanding of what it means for an algorithm to be recursive and for a recursive algorithm to be suitable to a given problem
- The ability to read, trace, and write simple to medium complexity recursive functions
- The ability to demonstrate an understanding of the relationship between recursion and looping
- An understanding of the definition of the “searching” problem and the “sorting” problem
- The ability to read, evaluate for correctness, and employ in different scenarios algorithms purported to solve the searching or sorting problems
- The ability to simulate the processes employed in binarySearch, insertion sort, selection sort, and merge sort.
- The ability to reason about the relative complexity of an arbitrary algorithm (not just constrained to searching/sorting)
From Test 3:

- An understanding of the interpretation of classes as user-defined types, specifying what values can be taken on by variables of the type and what operations are legal on variables of the type
- The ability to suggests choices for variables and methods and method semantics for classes appropriately based on textual descriptions of problems
- An understanding of the purpose of and the ability to design the five methods that I argue are important for any class: default (no parameter) constructor, copy constructor, destructor, assignment operator, and insertion (output operator)

- An understanding of the notion of encapsulation and data privacy and the mechanisms employed to support these ideas (separate header and source, public/private, friends)

- An understanding of the general properties of the LinkedList container, particularly as compared to the array container
- The ability to read and write simple to medium complexity LinkedList methods
- The ability to discuss how to/the appropriateness of employing LinkedLists in new scenarios

Specific things you do NOT need to know:

- Tail recursion (you won’t be asked to write tail-recursive algorithms; a tail-recursive algorithm could pop up by chance in a discussion of recursion, but the fact that it’s “tail-recursive” won’t be the focus of the problem)
- Recursion correctness verification
- Templates
- Stacks
- Makefiles
- Operator symmetry