Computer Science II — CSci 1200
Final Review

Overview

- Wednesday, May 9, 2007, 3:00-5:30pm Darrin 308.
- The exam is cumulative, and will include material from the last labs and lectures of the semester.
- You may take a one-page, double-sided, 8.5x11 crib sheet to the exam.

Important Topics

- Order notation
- Lists, vectors, strings, maps and sets
- Classes, including operators, constructors, and destructors
- Arrays, pointer and dynamic memory; pointer arithmetic
- Iterators and indices
- Problem solving
- Program design and choosing containers
- Recursion
- Merging
- Stacks and queues
- Linked lists
- Trees
- Hash tables
- Our own versions of the list, vector, string and set classes.
- Priority queues, heaps, leftist heaps
- Class hierarchies, inheritance and polymorphism

How to Study

- Redo homeworks, labs, class exercises, review questions and test questions from throughout the semester
- Make up and work through examples of heaps and leftist heaps
- Make sure that you understand class hierarchies, including the way constructors, destructors, virtual functions and polymorphism work.
- Make sure that you understand heaps, heap implementations as a vector and leftist heaps.

Practice Problems / Study Questions

There are no relevant practice problems from previous semesters' exams to help you with this material, so please review and redo lab and class exercises. The following questions (and answers) may help.

1. Show all binary heaps that can be created from the values 1, 2, 3, 4. Draw these as trees, but then write them as vectors. Show all leftist heaps that can be created from these values.
2. How can you efficiently merge two binary heaps? Why is it much easier / more efficient to do with a leftist heap?
3. What are the member variables of a derived class object? What about member functions?

Answer Summaries

1. There are three binary heaps. All of these are leftist heaps. There are three additional leftist heaps.
2. The simplest way to merge binary heaps is to concatenate the vectors and then run `make_heap`. This is an $O(n)$ operation. Merging leftist heaps is $O(\log n)$, and is easier because the trees are less rigidly structured and are formed to have empty locations where efficient merging is possible.
3. All member variables declared in the class itself and any other class above it in the hierarchy. The same is true for member functions except that when derived class functions have exactly the same declaration, the derived class functions take precedence.