Exam 2 Review

CS 3358
Summer I 2012

Jill Seaman

Exam 2

- Wednesday, June 27, 5:40pm to 7:00pm
- Derr 241 (here)
- Closed book, closed notes, clean desk
- 20% of your final grade
- I recommend using a pencil (and eraser)
- All writing will be done on the test paper I will hand out.
- No calculators.

Exam Format

- 100 points total
  - Writing programs/functions/code (50-60%)
  - Multiple choice
  - Fill-in-the-blank/short answer
  - Tracing code (what is the output), tracing sorts or sort operations
  - Finding errors in code (recursive functions)

Templates

- Why? What are they for?
  - Type independence, generic programming
- Templated Functions
- Templated Classes
  - Everything goes in the .h file
- Be familiar with the examples
  - MemoryCell, Vector
- Be prepared to write code, convert existing function or class to template

Stack ADT

- Know the operations, how they work
  - O(1): push, pop, isFull, isEmpty
  - makeEmpty
- Be able to implement an array or linked list version (singly-linked list)
- Be able to use a stack to solve a problem
- Be familiar with the sample code:
  - IntStack: the static stack class in the notes
  - stack_3358_LL.h: the linked-list implementation on the website
- Array vs Linked List implementations

Queue ADT

- Know the operations, how they work
  - O(1): enqueue, dequeue, isFull, isEmpty
  - makeEmpty
- Be able to implement a circular array (with wraparound) or linked list version (singly-linked list)
- Be able to use a queue to solve a problem
- Be familiar with the sample code:
  - IntQueue: the static queue class in the notes
  - queue_3358_LL.h: the linked-list implementation on the website
- Array vs Linked List implementations

Recursion

- How to write recursive functions
  - Base case
  - Recursive case (smaller caller)
- Recursion over non-negative ints and lists
  - arrays, vectors, linked list, List_3358, substr
- Know what's wrong with recursive Fibonacci
- Binary Search: understand recursive version
- You will be asked to write one or two recursive functions.

Sorting

- Understand the different sorts:
  - $O(N^2)$: selection, insertion, bubble
  - O(N log N): merge sort, quicksort
- Know the algorithms really well
  - Will not have to write code for an algorithm
  - May be asked to write psuedocode or give descriptions in English.
  - May be asked to show steps in the process (show result of a pass, or a merge, or a partitioning).
- Be familiar with runtime analyses and issues

Previous material

- Be able to apply to concepts covered in this exam:
  - Analysis of algorithms
  - Linked Lists
  - Dynamic memory allocation, the big three
  - Implementing ADTs
  - Using vectors, List_3358

Example Programming Problem

Given the ADT for the Stack_3358 at the end of the exam, implement the push, pop, isEmpty and isFull functions.

The class declaration would either:
a) include the private member variables or else
b) the question would state which implementation to use and you would provide the private member variables

Example "Tracing" Problem

Given the following array, what would be the contents
a) after the 4th iteration of the insertion sort?
b) after the 4th iteration of the selection sort?

```
3  7  2  12 56  1  42  9
```


Example Short Answer

What are the main steps of the merge sort? Include the base case and recursive case.

I will NOT provide the code for the sorting algorithms

How to Study

- Review the slides
  - understand all the concepts
- Use the book to help understand the slides
  - there will be no questions over material (or code) that is in the book but not on the slides
- Understand the code in the demo(s)
- Understand the homework assignment solutions
  - rewrite yours so it works (solutions on TRACS)
- Practice, practice, practice
- Get some sleep