Final Exam Review

CS 3358
Summer I 2012

Jill Seaman

Final Exam

- Friday, July 6, 8:00pm to 10:30pm
- Derr 241 (here)
- Closed book, closed notes, clean desk
- 30% 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

- 150 points total
  - Writing programs/functions/code (~40%)
  - Multiple choice
  - Fill-in-the-blank/short answer (big O functions)
  - Tracing code (what is the output),
    tracing sorts or sort operations, tree operations, heap operations
  - Finding errors in code (recursive functions)

Arrays, pointers, structs, objects, classes

- Know how to use vectors and strings
- Pointers, dynamic memory allocation and deallocation
- Structures, pointers to structures
- Shallow copy vs. deep copy
- Encapsulation, Information hiding, Interface
- Class declaration
- Default parameters, initializer list, const member function
- The big three (defaults, when to override)

Linked Lists

- How to define a linked list
  - Node definition
  - head (tail)
- Using null pointers
- Basic operations: be able to implement for single or doubly linked list.
  - constructor, append, insert, remove, destroy
  - display the list, copy constructor
- Know how to draw the lists
- Arrays vs. linked lists: pros+cons

Analysis of algorithms

- Understand the concept.
- Know the growth rate functions
  - Which ones are faster growing than others
- For a given algorithm/function, be able to do the runtime analysis (to say it is O(**F(N)**))
- Given two implementations, be able to say which is more efficient (faster)
- I will not necessarily give you the code this time, just a description of the algorithm.

Introduction to ADTs

- Data structure vs abstract data type
- Commonly used ADTs (list, set, bag, ...)
- Implementation vs. interface
- bag implementations
- List_3358, the cursor based list (demo+PA#2)
  - be familiar with the interface
  - be able to implement operations (array, linked list)
  - know the runtime analyses for the implementations we did

Templates

- Why? What are they for?
  - Type independence, generic programming
- Templated Functions
- Templated Classes
- Be prepared to work with templated classes and functions

Stack and Queue ADTs

- Know the operations, how they work
  - Stack: O(1): push, pop, isFull, isEmpty
  - Queue: O(1): enqueue, dequeue, isFull, isEmpty
- Be able to implement an array or linked list version (singly-linked list)
- Be able to use a stack or queue to solve a problem
- Be familiar with the sample code:
  - IntStack and intQueue with wrap (lectures)
  - Stack_3358_LL.h and Queue_3358_LL.h (website)
- Array vs Linked List implementations

Recursion

- How to write recursive functions
  - Base case
  - Recursive case (smaller caller)
- Recursion over
  - non-negative ints
  - lists: arrays, vectors, linked list, List_3358, substr
  - trees: Binary search trees
- You will be asked to write at lest one recursive function.

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
    - except main recursive function of quicksort or mergesort
  - Will 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

Hash tables

- Hash tables and (good) hash functions
- Collisions and collision resolution
  - Linear probing
    - Lazy deletion
    - Primary clustering
  - Quadratic probing
  - Separate chaining (pros+cons)
- Rehashing: how to expand the table
- Be able to hash a list of keys given a simple hashing function and collision strategy
  - Like the examples

Trees/Binary search trees

- Definitions and terminology, examples
- Traversals: preorder, postorder, inorder
- Binary tree
- Binary search trees
  - ordering property
  - ops: insert, remove, find, findMin, findMax
  - inorder traversal: sorted order
- Be able to implement the operations from PA7
- Be able to show (draw) tree after an operation

Heaps

- Understand the definition:
  - structural property: complete binary tree
  - ordering property: parent is smaller than children
- Array-based implementation
  - formulas to find nodes (children: 2i, 2i+1, parent: i/2)
- Operations
  - insert, findMin, deleteMin (percolate up and down)
  - understand algorithms (to trace, not code)
- Heapsort
  - understand the algorithm and runtime analysis

Example Programming Problems

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

Given the ADT for the BST_3358 at the end of the exam, implement the find and insert functions.

Know the programming assignments

Example Tracing Problem

- What is the inorder traversal for the following BST?
- What would the following heap look like after inserting 42?
- What would this BST look like after deleting 42?

A diagram containing a BST or heap
would be given for each question.

Example Short Answer

Give the runtime analysis Big O function for the insert operation in a doubly linked list when inserting before the cursor.

I will **NOT** provide the code for the operation

Answer would be something like: O(n) or O(1) or O(n^2) ...

What are the two main steps in the heapsort?

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 programming assignment solutions
  - rewrite yours so it works (solutions on TRACS)
- Practice, practice, practice
- Get some sleep