Final Exam Review

About this exam review

- I've prepared an outline of the material covered in class
  - May not be totally complete!
    - Exam may ask about things that were covered in class but not in this review session
    - I can't cover a ten week class in one hour!
  - Lacks detail
- This is an *interactive* review
  - This review is for ***your*** benefit!
  - If you have questions, ask them
  - If you don't understand something, say so

What will the exam format be like?

- Multiple Choice
  - Not necessarily easy
- Programming questions
  - Write this code (very brief)
  - What does this code do?
  - Where is the bug in this code?
- Explain this algorithm
  - Show how this algorithm works
  - How long does this take?
  - How could you do this better?
- What is this concept?
  - Differentiate from other concepts
  - Explain the concept

⇨ Questions similar to those on sample exams

What material will be covered?

- Final exam will be 200 points
  - Some multiple choice
  - Some short answer
  - There will be 1–2 bonus questions
  - 200 points means a bit less than 1 minute per point…
- Topics on the final exam may include
  - The book
  - Concepts covered in class
  - Programming assignments
    - Algorithms used
    - Tools used (only the required ones)
  - Java programming languages (Not C)

Java Programming

- What is an algorithm?
- Software Life Cycle
  - Problem Analysis
  - Design
  - Implementation
  - Test
  - Modification
- Fundamentals
  - Lexical elements
  - Comments
  - Keywords
  - Identifiers
  - Literals
  - Operators and punctuation
  - Data Types and Variables
- Methods
  - Calling and passing parameters
- Numeric Types
  - byte, short, char, int, long, float, double
- Arithmetic Expressions
  - Operators
  - Precedence and associativity
- Statements and Expressions
  - Empty, block,
  - Boolean expression, relational operators, comparisons

Java Programming (cont.)

- Conditional Statements
  - if
  - if-else
  - switch
    - case
  - while
  - for
  - break and continue
- Structured Programming
  - Top-down design
  - Method definition
  - public static void main(String[] args)
  - return
  - Scope of variables
  - Recursion
- Arrays
  - Declaring
  - Using
  - Initializing
  - Copying
  - 1D, 2D, 3D, …
- Data Abstraction
  - Classes define types
  - Encapsulation
  - Data hiding
  - public vs. private
  - Constructors
  - static vs. not static
  - Passing objects: references
  - final

O (n^2) sorting algorithms

- General concepts
  - Understanding sorting
  - O(n^2) sorts often have
    - Two nested loops
    - Simple code
- Insertion sort
  - For each element in the unsorted set, put it in the right place in the sorted set
  - Outer loop is over elements in the unsorted set
  - Can be fast if elements are near their “correct” position
- Selection sort
  - For each position in the sorted set, find the appropriate element in the unsorted set
  - Outer loop is over positions in the sorted set
- Bubble sort
  - Loop through set exchanging adjacent elements
  - Repeat until set is sorted (no exchanges take place)
  - May be fast if set is nearly sorted to begin with

Divide & conquer sorting algorithms O(nlogn)

- Mergesort
  - Divide array in half
    - Copy into new array
  - Sort each half recursively
  - Merge the two halves into the original array
- Performance is
  - O(n log n)
  - Uses lots of memory: O(n^2 log n)
- Advantages
  - No worst case performance
  - Easy to write
- Disadvantages
  - Memory usage

- Quicksort
  - Pick a pivot element
  - Divide array into two halves: less than pivot and greater than or equal to pivot
  - Recursively sort each half
- Performance is
  - Usually O(n log n)
  - Worst case O(n^2)
    - Prevent this by picking pivot intelligently
    - Pick median of 3–5 randomly chosen elements
- Advantages
  - Sort in place (no extra array)
  - Easy to code

Non-tree Data Structures and Concepts

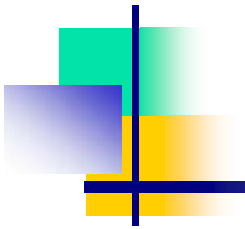
- Linked lists
  - Singly linked lists
  - Doubly linked lists
  - Operations on linked lists
    - Insert
    - Lookup
    - Delete
  - Keeping linked lists ordered
  - Running time
- Stacks
- Queues
- Linear Search
- Binary search
  - How it works
  - How long it takes

Exceptions

- Occur when something “unusual” occurs
  - Must have a way to signal calling procedure
  - Can’t always use return value to do this
  - Use an exception (in Java)
- Possibility declared with “throws”
- Exception generated by
  - Creating an Exception objecting
  - Using a “throw” statement
- Exception caught by “try … catch” statement
- Exceptions don’t exist in C
  - Instead, return “illegal” value
  - Not always possible…

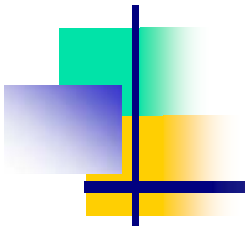
Binary trees

- Representing binary trees
  - Node structure
  - Storing data in trees
- Operations on binary trees
  - Insertion
  - Lookup
  - Deletion
- Describing binary trees
  - Size
  - Depth
  - Full binary trees
  - Balanced binary trees
- Traversing binary trees
  - Pre-order
  - In-order
  - Post-order
- Binary trees and files
  - Writing trees to files
  - Reading trees from files
- Using binary trees
  - Binary search trees
  - Huffman decoding (Assignment #4)
- Running time of binary tree operations
  - Insertion: O (log *n*)
  - Lookup: O (log *n*)
  - Deletion: O (log *n*)



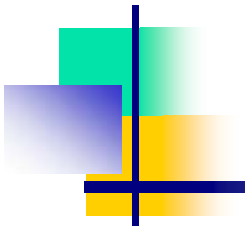
Trees

- Binary Search Trees
  - Sorted order
- Balanced Trees
- 2-3 Trees
- 2-3-4 Trees
- Red-Black Trees
- AVL Trees



Hash Tables

- Hash functions
- Operations
  - Insert
  - Lookup
  - Delete
- Metrics
  - How full?
- Collision resolution
  - Chaining
  - Rehashing
- Running time



C programming language

- Understand the difference between C and Java
  - References vs. pointers
  - Garbage collection vs. freeing memory explicitly
  - Classes (object oriented) vs. declarative
- Understand how C programs are structured
  - Header files
  - References to external functions
- Understand C syntax and concepts
  - Pointers
  - Arrays
  - Structures

How should I prepare for this exam?

- Understand the concepts we've covered in class
  - ***Don't memorize***
    - You should be able to rederive a result without memorizing the answer
    - Understanding concepts will (hopefully) prevent answers that are obviously wrong
  - Focus on understanding the *how* and *why*
- Know how to use Java well, understand C
  - Difficult things like objects (Java), pointers & arrays (C)
  - Understand syntax
- On the exam
  - Do the questions you know first: they may take less time than the points would indicate
  - Make *educated* guesses if necessary