Midterm 2 Topics:

The second midterm can be taken during any 2-hour block April 30-May 1 (Thursday and Friday). If you need an extension beyond these days, let me know by Tuesday at the very latest. Midterm 2 is implicitly cumulative in the sense that many of the concepts build on each other, but overall it will focus more on the second half of the semester (in-class material days 15-26, labs 4-7, and reading weeks 8-13). You may use a 1 page (front and back), hand-written "cheat-sheet" (created by *you*), but no other notes or resources. (You will not need a calculator.) I have put vocab in blue.

1. Queues (FIFO - first in first out)
    - Queue ADT (Abstract Data Type ≈ interface)
    - Queue implementations and their pros/cons/runtimes: circular array, list
    - Deque ADT and implementation
    - Application of stack/queue: Shunting-Yard algorithm
2. Graphs (collection of vertices and edges)
    - General graph terminology
    - Graph ADT
    - Implementations and pros/cons/runtimes: Adjacency List, Adjacency Matrix
    - Application: Kruskal's algorithm for minimum spanning tree (basic idea)
3. Binary Trees
    - Tree vocabulary (parent, child, depth, etc)
    - Motivation for trees: binary search (recursive method)
    - Binary Search Tree (left child less than parent, right child greater than parent)
    - Runtime of insertion; use of binary trees to speed up insertion sort
    - Implementation: recursive data structure
    - Traversals and recursive implementation: in-order, pre-order, post-order (all depth-first)
    - Application: Huffman coding algorithm (basic idea)
    - Negatives of having an unbalanced tree + basic idea of tree rotations to rebalance
4. Priority Queues and Heaps
    - Priority Queue ADT (idea: want to remove object with top priority each time)
    - Pros/cons/runtime of using sorted vs. unsorted list
    - Heaps as a semi-sorted data structure that implements the priority queue interface
    - Implementation of a heap using an array (breath-first ordering)
    - Heap runtime guarantees and use in heap sort

5. Maps and Hash Tables
   - Map ADT, idea of key-value pairs
   - Hash tables (i.e. dictionaries) as an implementation of the Map ADT
   - Using hash functions to find the desired position of a key
   - Collision handling methods (probing vs. chaining)
   - Linear probing implementation of a hash table
   - Goal of $O(1)$ for lookup and insertion
   - Hash sets are hash tables with keys only (no values)
   - Overriding `.equals(Object o)` and `.hashCode()`
6. Sorting algorithms (including runtimes, implementation details, and in-place vs. out-of-place)
   - Insertion sort (using a list vs. binary search tree)
   - Heap sort
   - Quick sort
   - Radix sort
   - Merge sort
7. Concepts from the first half that are still heavily used
   - Arrays, Array Lists, Linked Lists
   - Class structure (instance variables, methods, constructors, getters, setters)
   - Primitive types (e.g. `int`, `double`, `boolean`, `char`) vs. Objects (e.g. `String`, `ArrayList`)
   - Interfaces
   - Generics
   - Big-O notation to describe runtime