EE 200 Final coding interview study sheet

11. Object-oriented programming

- Write code for a class, using the following features:
  - access control
  - overloaded functions
  - default arguments
  - overloaded operators
- Write code containing references
- Explain the difference between a reference and a pointer, in terms of declaration and usage

12. Constructors and destructors

- Write constructors, including copy constructors
- Use initializer lists to initialize class members, and explain the advantage of doing so
- Use `new` and `delete` to manage memory
- Read a short piece of code (5-10 lines) and identify how many instances of an object will be created

13. Templates

- Instantiate and use a template class (e.g., from the standard template library)
- Write a templated function
- Write a templated class
- Explain why template code is usually placed in a header file rather than a `.cpp` file.

14. Inheritance

- Identify whether inheritance is appropriate for a given scenario
- Describe the difference between static dispatch and dynamic dispatch
- Use `virtual` to write polymorphic functions
- Define an abstract class, and explain why such a thing is useful

15. Error handling

- Write code to catch and handle exceptions
- Trace what happens when an exception is thrown, in terms of the call stack, try/catch blocks, etc.
- Determine what level of exception guarantee a function makes (no throw, strong, basic, none)
- Explain the reasoning behind the rules in Section 19.6 (“Using Exceptions Properly”) of AoP

18. Sorting and Big-O

- Given the description of an algorithm, find the big-O run time.
- Explain the difference between average, best-case, and worst-case run time.
- Give the big-O runtimes for Quicksort, Mergesort, and Heapsort, and explain when you would choose one over the other.

19. Linked lists

- Draw a diagram of a linked list and explain how insertion and removal work in terms of the pointers and values.
- Give the big-O runtimes for linked-list operations: insert, remove, lookup by index, lookup by value.
- Explain when you would use a linked list versus an array.

20. Binary search trees

- Draw a diagram of a BST and explain how insertion and removal work in terms of the pointers and values.
- Give the big-O runtimes for BST operations: insert, remove, lookup by value.
- List advantages and disadvantages of a BST compared to a linked list or array.
- Given a simple definition for a BST class, write a recursive function to perform preorder, inorder, or postorder traversal of the BST.

21. Hash tables

- Outline the steps requied to add an element to a hash table.
- Give the big-O runtimes for hash table operations: insert, remove, lookup by value.
- List advantages and disadvantages of a hash table compared to a binary search tree.
- Explain what makes a good hash function.
- Give at least two examples of why a hash function is useful besides a hash table.

22. Heaps

- Give the big-O runtimes for heap operations: push, pop, and peek (i.e., `top()`)
- List advantages and disadvantages of a heap compared to a linked list, array, or BST.
- Write code to perform a re-heaping operation after adding or removing an element.