CS5001
Fall 2023
Final Exam Study Sheet

Format:

- Part I: Short Answer & Multiple Choice
- Part II: Understanding Code and/or flowcharts
- Part III: Writing Code

Covered topics:

- Classes and objects
- Functions as Objects
- Data Structures, Stacks
- Algorithmic Complexity
- Lists, Strings, and Dictionaries
- Unit testing
- File processing and Exception Handling
- Recursion

To study and review

The best way to study for a computer science exam is to re-do and review previous material. Look at worksheets we've done in class, come up with the answers from scratch, and verify to make sure you're right. Look at the quizzes we've done, come up with the answers again, and verify to make sure you're right. Re-do short labs and practice problems, or do practice problems for the first time.

During the exam

Something is better than nothing -- that's always my philosophy on exams. We give partial credit. For this exam, you'll be writing code, so make sure you submit **code that works** even if it doesn't run perfectly for the problem. We'd rather see your code fail a few tests than crash in a big horrible error.

Classes and Objects

- You should know that a class is a design and an object is an implementation of that design
- Expect to implement a constructor `__init__()` for a class, with or without parameters. Know how to implement the __str__() and __eq__() methods as well. Prof K may not ask for them, but you need to be prepared to do so if he does
- You should know that a function attached to a class is called a method, and that it always takes *self* as its first parameter.
- Given a class definition, you should be able to create objects and call their methods.
- Given a driver that creates objects and calls methods, you should be able to implement the methods it's calling.

- You should remember that functions are first-class objects. We can pass functions to other functions (conceptually, these are higher-order functions). Be sure you understand how to both pass a function and receive a function as a parameter. Remember that the function call operator () is what invokes the function. The name of a function is the address.

Data Structures, Stacks

- You should know why we define data structures beyond the most basic one, the list
- You should know that there are four basic stack methods: `push`, `pop`, `peek`, and `is_empty`.
- You won't be asked to define a Stack class, but you will be asked to use one that already exists. Be ready to create Stack objects and to invoke those four methods to solve a problem.

Algorithmic Complexity

- You should know why this topic is important (i.e., why computer scientists care about efficiency so much) and that we measure efficiency independent of programming language, processor speed, memory size, etc.
- If you've written code to solve a problem, expect to identify the upper-bound on its run-time complexity (in big-O notation).
- You'll be asked to rank complexity classes relative to one another. From most efficient to least efficient, they are: $O(1)$, $O(\lg n)$, $O(n)$, $O(n \lg n)$, $O(n^k)$, $O(k^n)$, and $O(n!)$.

Lists, Strings, and Dictionaries

- Lists and strings have been covered pretty well on quizzes and the midterm. They will certainly be on the final as well, especially under the topic of run-time complexity, because we identified algorithms that work on lists.
- As on the midterm, if we ask you to write a function that works on a list, we'll usually specify that you can't use Python shortcuts or certain list methods. Be sure you can iterate through a list by value (`for element in mylst...`) and by position (`for i in range(len(mylst))...`).
- Expect to see a problem where you earn more points for an efficient solution, and fewer points for a correct but less-efficient one.
- You should be able to initialize a dictionary with specified key/value pairs.
- You should be able to add a key/value pair to an existing dictionary.

Unit Testing

- You won't need to write unit tests or use the TestCase class on the exam, but you do need to read and understand them.
- We'll test all the functions you write for the exam using Python's unittest.
- Prof K may ask you to specifically create a class and/or a set of functions that pass a particular unit test he gives you. So, from a given set of unit tests, ensure you know how to write a function that will pass the tests.
-

Recursion

- Expect to write or trace at least one (simple) recursive function.
- It will either process a list (e.g., summing the elements of the list), or it will work on a numerical input (e.g., recursive factorial).