ECS 32B: Exam #1 Details

Instructor: Aaron Kaloti

Summer Session #1 2020

Contents

1 Changelog

You should always refer to the latest version of the syllabus.

- v.1: Initial version.

2 Exam Format

This exam will be a timed Gradescope quiz that will be proctored over Zoom. While in the Zoom meeting, you are required to have your face visible through the webcam at all times. **If you fail to meet such requirements, e.g. you don't join the Zoom meeting or you leave your webcam off even after I try to tell you to turn it on**, then I may have to give you a zero on the exam. The purpose of proctoring through this Zoom meeting is to ensure that it is truly you who is taking the exam and that you are not communicating with any other student as you take the exam. The Zoom meeting will be open at 9:55 AM. The exam (i.e. the Gradescope quiz) should become available around 10:00 AM. Once you start it, you will have 70 minutes to finish. (Thus, if you start at 10:02 AM instead of 10:00 AM, you will still have 70 minutes.) If you start the exam after 10:10 AM, then I expect some sort of explanation over email; if you do not provide one ASAP (as in, around the time you start the exam), then I may have to reject your exam submission.

Some of the questions on the exam may ask you to write code and encourage that you use some Python development environment of your choosing (e.g. Python IDLE, Mu editor, PyCharm) as you do so. On such questions, I will permit you to submit a Python file for that question instead of pasting your code into the answer box. The reason for this is that sometimes, students have trouble pasting code into the answer box on Gradescope quizzes, perhaps due to a bug on Gradescope's end. In case you wish to test the functionalities of the Gradescope quiz, I have created a sample Gradescope quiz called "Exam File Upload Test" that will always be open.

All audio will be muted during the meeting. This means that you can use headphones or ear buds and listen to music, if you want. I am allowing this because I know that for some students, they may not have access to a quiet environment or control over how loud the others they live with are. Please do not use any of the disruptive features such as the "Raise Hand" feature during the Zoom meeting; if you need my attention for whatever reason, use email, as I will constantly check my email during the exam.

The exam is open note, open keyboard (since you need to type your answers somehow), etc. I might also recommend that you have scratch paper ready, just in case. **You are not allowed to search for answers online. You are not allowed to ask each other for help. Cheating on an exam will be caught and punished, almost certainly with an F in the course.** You are allowed – and for some problems, *advised* – to use an editor such as the Mu editor or Python IDLE, so do make sure that you have an editor that you are comfortable with. You are ***not*** allowed to consult me or the TAs for hints on the correct answers, but you *are* allowed to ask *me* for clarifications on the questions, over email.

If you finish the exam early, leave the Zoom meeting; no need to let me know. Zoom generates a log of all of those who enter and exit the meeting.

If you want, you can set a virtual background on Zoom.

If you are having connection issues due to being in the Zoom meeting, please shoot me an email immediately (yes, during the exam).

If you are a student who has accommodations that you have informed me about, then I will contact you soon about how the exam might be handled differently in your case. Please email me if I do not contact you soon.

2.1 Regarding Specific Exam Problems

The exam will be a mix of conceptual questions, small coding questions that may ask for very small snippets of code (i.e. not more than 1-3 lines, perhaps), and larger coding questions that ask you to write a function or program.

The larger coding questions are meant to evaluate a baseline level of competence with the ECS 32A concepts that are required for this course. As stated in a Canvas announcement, I have posted my ECS 32A lecture slides to Canvas, just in case. The coding questions could involve topics such as conditional statements, loops (including `break` and `continue`), strings, lists, dictionaries, and tuples. They could also involve sets. The larger coding questions will *not* involve file I/O, user-defined classes, references, or unit testing. However, the conceptual questions or small coding questions could involve these topics (except unit testing, which – as stated below – will not be tested).

You should make sure that you understand all of the solutions to conceptual HW #1.

You may have to type some math in some of the questions. In such cases, there is not a strict expectation as to how you format certain mathematical operations (e.g. you could denote exponentiation with either `**` or `^`), so long as your answer is unambiguous. For denoting big-Theta or big-Omega, you can use Theta(...) or Omega(...). There will be a reminder about all of this on the exam.

I will probably not post practice questions, especially given that conceptual HW #1 already serves as practice questions for many of the concepts.

3 List of Topics

Below are a list of topics that you should make sure that you understand before taking exam #1. Not all of these topics will appear on the exam.

3.1 Slide Deck #1: Computational Complexity

- Big-O. Formal definition. Proving that a big-O holds. Big-O is an upper-bound.
- You will not be asked to create a cost function.
- Definitions of big-Ω and big-Θ. You will not be asked anything about proving big-Ω or big-Θ.
- Be able to identify the worst-case time complexity (with each of big-O, big-Ω, and big-Θ) of a given segment of code or function.
- Common categories of functions: constant, logarithmic, polynomial, exponential, factorial. You will not be expected to tell if a given segment of code runs in exponential or factorial time. However, you may be expected to tell if a mathematical function is big-O (or big-Ω or big-Θ) of an exponential or factorial function (without having to prove it).
- Space complexity. You will only ever be asked about auxiliary space.
- Best-case time/space complexity and average-case time/space complexity will not be tested.
- You do not need to understand the different notations for saying that a given function $T(n)$ is $O(f(n))$.
- You will not be tested on the symmetric or transpose symmetric properties of big-O and the like.

3.2 Slide Deck #2: Python Concepts

- Exceptions (everything in the slides about them).
- File I/O (everything in the slides about them).
- User-defined classes.
- References. `is` operator. Shallow copying vs. deep copying. Why deep copying is needed.
- Sets.
- Unit testing will not be tested on this exam.

3.3 Slide Deck #3: Searching

- Linear search.
- Binary search. How the algorithm works. You do not need to understand its worst-case time complexity or space complexity.

UCDAVIS
COMPUTER SCIENCE