ECS 32B Syllabus

ECS 32B Introduction to Data Structures
Fall 2020

Instructor

Jiaming Jiang
jmjjiang@ucdavis.edu

Teaching Assistants

David Bauer (davbauer@ucdavis.edu)
Siena Saltzen (scsaltzen@ucdavis.edu)

Lectures

MWF 3:10 – 4:00pm
Due to the pandemic, live lectures will be given via Zoom.
Zoom Link for the lectures: https://ucdavis.zoom.us/j/98540332478
Passcode: 693457

Discussion Sections

Note: no discussion sections during the first week of classes
Due to the pandemic, discussion sections will be held via Zoom using the same link as lectures.
R 8:00 – 8:50am

Office Hours

Jiaming: MTWRF 11:30am – 12:30pm (Zoom Link: https://ucdavis.zoom.us/j/99830324364)
David: MTWRF 10:00 – 11:00am (Zoom Link: https://ucdavis.zoom.us/j/97497634668?pwd=dkgrVXRaS0tjVjRpVmtKeTE3ajBiQT09)
Siena: MTWRF 6:00 – 7:00pm (Zoom Link: https://ucdavis.zoom.us/j/98560689484?pwd=cmx3TFkxQWhtRUdLU2I3ZUJYdENLdz09

Objectives

- Students will learn to think clearly about and solve complex and poorly-defined programming tasks.
- Students will be able to (a) find appropriate abstractions to solve a complex

problem, (b) choose appropriate data structures and algorithms, (c) analyze simple algorithms and discuss tradeoffs among data structures, and (d) get it all working.

Prerequisites

ECS 10 with a C- or better or ECS 30 with a C- or better or ECS 32A with a C- or better or ECS 36A with a C- or better

Credit Restrictions

No credit to students who have completed ECS 36C or ECS 60 or higher

Textbook

Problem Solving with Algorithms and Data Structures Using Python (3rd edition) by Bradley N. Miller and David L. Ranum. The book is available online at no cost here: https://runestone.academy/runestone/books/published/pythonds/index.html

Tentative and Approximate Schedule

Week	Topics	Chapter Reading	HW
1	Introduction and Algorithm Analysis	1.1 – 1.3, 3.1 – 3.4	
2	ADTs, OOP Basics	1.4 – 1.8, 2	HW 1
3	Linear Data Structures	4	HW 2
4	Recursion	5	HW 3
5	Searching	6.1 – 6.4	HW 4
6	Sorting	6.6 – 6.12	HW 5
7	Hashing	6.5	Midterm, Bonus HW
8	OOP Advanced, Trees	7.1 – 7.7, 7.11 – 7.22	HW 6
9	Heap and Priority Queues (Thanksgiving)	7.8 – 7.10	HW 7
10	Graphs	8	
11	Graphs	8	HW 8

Grading

- 8 homework assignments: each is worth 7%; total is 56%
- 1 midterm exam: 20%
- 1 final project: 24%
- Bonus homework assignment: 5%

Exam and Project Dates (midterm dates subject to change based on class progress)

Midterm: Friday, November 13

Final Project: Friday, December 18, 5pm

Late Homework and Missing Exams:

We do not accept late homework except in the case of documented illness, documented family emergency, or documented university-approved absence. Forgetfulness is not an accepted excuse for late homework. Submitting your homework to the wrong repository on Canvas, even if submitted before the deadline, is not an accepted excuse for late homework. Also, a note stating only that you visited the student health center is not sufficient documentation of illness.

There will be no makeup midterm exams. But there is one bonus homework assignment as extra credit. We will also incorporate extra credit in the final project.

Submitting Homework for Evaluation:

- Your solutions to homework assignments must be submitted through **Gradescope** by the indicated day and time.
- **For programming questions** in the assignments, the autograder of Gradescope will evaluate your code on demand. You can submit as many times as you want. For more information, please see Home - Gradescope Autograder Documentation
- **For non-programming questions** in the assignments, you may either hand-write or type the solutions. But the submitted files must in PDF.
- Any programs must be written using **Python 3**. Programs written in earlier versions of Python, or in any language other than Python 3, will not receive credit.
- Submit your solution early, in case of problems. Don't wait until the last minute. Make sure you submit the correct file. We won't accept late homework, even if you had difficulty with Gradescope or you turned in the wrong thing.
- *We do not accept e-mailed homework.*

UC Davis's Code of Academic Conduct

You are expected to have read UC Davis's Code of Academic Conduct, which you may find here: http://sja.ucdavis.edu/files/cac.pdf

If you have questions about the Code of Academic Conduct, please ask your instructor or contact the Office of Student Support and Judicial Affairs.

Additional Thoughts about Academic Misconduct:

As the instructor and teaching assistants, it is our responsibility to make tests and assignments that are fair, to grade fairly, to look for cheating, and to refer students who cheat to Student Judicial Affairs for possible sanctions. Students in ECS classes are most often referred to SJA for copying each other's programs or copying on tests.

As students, it is your responsibility to avoid cheating and to discourage other students from cheating.

It is pretty clear what it means to cheat on a test, in this class just like any other. It is sometimes less clear to students when they are cheating on a programming assignment. We want you to help each other, and we want you to look at examples of similar programs. So how do you know when helping and looking crosses the line into cheating? Here's our basic rule:

You should write each line of your program yourself, and you should know what it does and why it is there.

To check yourself, you should type each line of your program yourself; *never copy or cut-and-paste from another file*. While you're typing, ask yourself, do I know what this line does? Why am I including it in this program?

Often the easiest way to write a program, even in industry, is to take an already-existing program that does something similar, and change it around. This is fine in "real-life", but in this class it is better not to cut-and-paste whole programs or even single lines, since as a beginner you need to concentrate on every line. You should, however, look very carefully at the example programs from lecture or the textbook, and figure out how your programs should be similar or different.

Writing programs can be very, very frustrating. Sometimes you don't know how to start. Sometimes your program seems perfect, but it doesn't do what you think it should be doing. We want you to talk to the other students, to friends who are programmers, to the TA, to the instructor, to anyone who can help! We want you to show them your programs and ask them what's wrong. But make sure when the conversation is over that you understand every line of your (hopefully improved) program. If your friend is telling you exactly what to type, you are cheating.

If you are looking at another student's program to help him or her, that is not cheating. If you are showing another student your program to help him or her, that is cheating. If you are looking at another student's program while typing in yours, or if you cut-and-paste

from another student's program, that is obviously cheating and you are very likely to be caught. Two (or more) students who work together on a single solution and then submit separate copies of that solution have crossed the line into cheating. A student who copies a solution that was provided by a tutor or that was found on the Internet has crossed the line into cheating.

You will quickly see that there are always many, many ways to write a program for a particular programming assignment, just like there are many ways to write an essay for a particular English assignment. And it is almost as easy to recognize two almost identical programs as it is to recognize two almost identical essays. For some of the assignments in this course, we will use software to detect almost identical programs. If you find yourself changing around someone else's program to try to fool the detection software, you are cheating.

One other thing: a given homework assignment is worth only a small fraction of your final grade. UC Davis's Code of Academic Conduct gives faculty the authority to assign a final course grade of "F" in cases of academic misconduct. So, when you cheat on the homework, you're risking a failing grade in the course (and maybe worse) for that relatively small fraction of your grade. You need to ask yourself if that's a risk worth taking.