Course Information

You are responsible for everything on this handout. Read it.

Lectures

TR 1:40–3:00 in 205 Olson. The CRN for this class is 59644.

Discussion Section

F 1:10–2:00 in 205 Olson.

Instructor

Phillip Rogaway
Engineering II, #3063
Phone: 752–7583
`rogaway@cs.ucdavis.edu`
http://www.cs.ucdavis.edu/~rogaway/

Office hours: Tuesday 5:10-6:30 (overlaps with UG advising); Friday 10:10-11:00; Saturday 5:00-6:00. Office hours subject to occasional cancellation; watch the newsgroup.

Teaching Assistant

Glen Nuckolls
`nuckolls@cs.ucdavis.edu`
Office hours: Wednesday 2:10-3:00 and Friday 3:10-5:00 in Engineering II, #3090.

Grader

Micah Giddens
`giddens@cs.ucdavis.edu`

Grading

Weekly problem sets — 25%
Midterm — 25%
Pop quizzes — 8%
Final — 42%

To pass the class you must pass the final and attempt the bulk of the homeworks.

The final is exam group M: Tuesday, March 23, 8–10 am. Closed book, closed notes.

There will be three quizzes, each 10-15 minutes. They might be unannounced, or they might be announced during the preceding class. It is possible that not all the quizzes will be counted. If you miss a quiz, there is no making it up.

It is possible that the grader will only grade a proper subset of the problems turned in.

If you think a homework or quiz problem was misgraded speak first to the grader and then to me or to Glen. If you think a midterm question was misgraded speak to me or to Glen. If something was just recorded wrong or summed up wrong then Glen will handle it. Misgrading concerns must be resolved within one week.

Prerequisites

ECS 20 (formerly ECS 100) is the prerequisite for this class (with Math 108 recommended). This is a serious prerequisite, in the sense that you will not do well in this course—you might not even pass—if you do not have the mathematical maturity consistent with having taken, and understood, ECS 100. In particular, you need to be able to understand and create proofs, and to distinguish a real argument from a bogus one. While I expect very little in the way of *particular* mathematical background, I do expect that sort of mathematical maturity. If you are a CS major who has trouble with math, consider taking some or all of your math electives before taking this class.

Text

Michael Sipser, *Introduction to the Theory of Computation*, PWS Publishing Company, 1997. We will cover most of the material of Chapters 0–5, 7.

There is now a second printing which corrects a myriad of typos.

Course Web page

http://www.cs.ucdavis.edu/ rogaway/classes/120/winter99/

Access the page directly (you might want to set a bookmark), or navigate from my own home page. Visit the page regularly. Retrieve homeworks, homework solutions, and other course materials from this page. Most materials will be PostScript, so you may need to download a PostScript previewer and configure your browser to invoke it.

Course newsgroups

`ucd.class.ecs120` — This is for me or Glen to communicate things of general concern to the class (most typically, correction to any homework problems found to be in error). Please read this newsgroup regularly; you are responsible for anything on it. Don't post to this newsgroup. Use the following newsgroup instead.

`ucd.class.ecs120.d` — This is the "discussion" newsgroup which you may use to communicate among yourselves, and also to ask questions directed to other students at large, me, or Glen. Read this newsgroup or not — the choice is yours. Do not post what amounts to solutions to homework problems to the newsgroup.

Problem sets

Homeworks will usually be due on Mondays at 4:10 pm. Turn in homeworks in the marked box in Engineering II, #086. No late homeworks are accepted.

Much of what one learns in this course comes from trying to solve the homework problems, so work hard on them. I intend for you to find some of the problems challenging. Doing a conscientious job on the homeworks is the best preparation for the exams, and it is essential for mastery of the material.

Your writeups should be clear, terse and neat. Aim for elegance. Understandability of the solution is as necessary as correctness. Don't be shocked to lose points if you provide a correct solution with a not-so-good writeup. As with an English paper, don't turn in a first draft; it takes refinement to produce a good writeup.

If you can't solve a problem, briefly indicate what you've tried and where the difficulty lies. Don't try to "fake" a solution. Know what you know and be clear about it.

Oddly (to me!) many students are happier to spend long hours hacking in front of a machine than peacefully thinking beneath a tree. Try not to be that way. All I can tell you is that as programming endeavors become complex the bulk of your time in that domain, too, should be spent in contemplation. The skill you learn in this class are more applicable than you might imagine.

Collaboration

I discourage collaboration on homeworks. When I ask people who have are good at this subject if they every worked with anyone on their equivalent of this course, the answer is always no.

If you do collaborate, the manner in which you collaborate will have a profound impact on how much you get out of the homeworks (and this, in turn, will have a big impact on how you do on the exams). First, think about the problems and try to solve them on your own. If, after giving a problem some real though, you just can not solve it, then you might wish to discuss it with other students, with me, or with the TA.

Academic misconduct

If you discuss problems with anyone, make certain to acknowledge him/her/them. Also acknowledge any books which you consulted other than your own.

Write up problems entirely on your own (even if you discuss a problem with someone else).

Some homework questions will have been used in prior years (either by me other professors). Do not consult old problem set solutions for this class — neither solutions of another student from a prior term, nor solutions handed out in class from a prior term.

If you are having personal or academic problems which are motivating you towards academic misconduct, come talk to me or Glen. We are not monsters. (The recent film on space-alien teachers who kill and eat their students is, in my opinion, somewhat exaggerated.)

Some hints

Most students find this class abstract and challenging. For some it is the hardest undergraduate class they take.

I really want you to think. Don't try to solve the problems by doing some sort of "pattern matching." It won't work.

This course is about learning a certain sort of problem-solving skill at least as much as it is about learning the specific material. Keeping this in mind may help put things into better perspective.

You might find any of the following hints helpful.

More than with other courses, you must not get behind. What we will do will keep building on what we have already done. Don't lose the thread. If you get seriously behind you will probably find it impossible to get back on track.

Do the reading before each lecture. The next reading will be on the Web page and indicated in class. Make sure you can do all the book's exercises. (The book distinguishes "exercises" and "problems".)

Be selective in note-taking. Often it works better to just sit there and listen. If you feel you need comprehensive notes, team up with someone else so that you only sometimes have to scribe notes.

If you get involved in a study group, don't let the emphasis degenerate into collective attempts to solve the homework problems. You'll learn more struggling on your own.

A remark for CS majors

This class used to be required, but now you have an option of taking ECS 122B, a course which is less abstract and less fundamental. ECS 122B is offered this term. But if you're bound for graduate-school, I suggest you to take ECS 120.

A remark for ECS majors

The requirement is this course or ECS 122A. Good students and students bound for graduate-school are well-advised to take both. For students who will only take one, I usually recommend it be ECS 122A.

Parting thoughts

This is my favorite course in the ECS curriculum. We really get at the question of what *is* computation. What could be more interesting or more fun?