Course Information

You are responsible for everything on this handout. Better read it!

Time and Place

Lecture are MWF 9:00–9:50 in 107 Cruess. (Where's that!?) Discussion section is M 4:10-5:00 in 147 Olson.

Instructor

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

Office hours: Tuesday 9-10 and Friday 10-11:20 (overlaps with UG advising);

Teaching Assistant

Glen Nuckolls
`nuckolls@cs.ucdavis.edu`
Office hours: to be announced (see course Web page).

Grader

Micah Giddens
`giddens@cs.ucdavis.edu`

Grading

Weekly problem sets — 25%
Evening Midterm — 25%
Three pop quizzes — 10%
Final — 40%

This class has an **evening midterm**. It will be on **Tuesday, May 11, from 7 – 9 pm**. I use an evening midterm because when I give a 1-hour exam in this subject many students feel time-pressured. With the evening format you should have plenty of time. If you have a time conflict come to speak to me by April 23 to arrange an alternative exam time.

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

All exams are closed book, closed notes.

There will be three quizzes, each 10-15 minutes. They will not be announced. 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 feel certain that a homework or exam problem was misgraded you may submit it for regrading. This must be done within one week of when the material was returned to you. If something was just recorded wrong or summed up wrong then just show it to me and I will fix it.

Prerequisites

ECS 20 (formerly ECS 100) is the prerequisite for this class. 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 meaningful proof 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 Chapters 0–5 and 7.

The second printing of this text corrects a myriad of typos.

Course Web page

http://www.cs.ucdavis.edu/˜rogaway/classes/120/spring99/

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. You will even be able to check your grades from here. 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.

E-mail questions

I don't normally respond to questions on course material sent to me by e-mail. There's just not enough time. Please see me in office hours instead. Or use the newsgroup.

Problem sets

Homeworks will be due at 4:10 pm. Turn in homeworks in the marked box in Engineering II, #2131. 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. 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 permit, but strongly discourage, collaboration on homeworks. I think that solitary struggling on homeworks is part of the joy and the value of this class.

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 thought, you just can not get anywhere with it, then you might wish to discuss it with other students, with me, or with the TA. In any case, you must individually write up all solutions.

Many students get through the homeworks by forming study groups and jointly trying to solve the homeworks. This is permitted, but evaluate carefully if you are learning the desired problem solving skills better than if you were spending the same time trying to get it on your own. The goal of homeworks is less to maximize your points than to learn a certain set of skills.

Academic misconduct

If you discuss problems with anyone, please be 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). Don't look at other peoples' solutions. Highly similar writeups will be considered cheating.

Some homework questions will have been used in prior years. 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 your homework resembles the grader's solution he's bound to notice, and you'll have a problem you don't want to have.

Some hints

Most students find this class abstract and challenging. For some it will be the hardest undergraduate class they take. But don't be scarred; you really can succeed in this class, and it's not that hard to do. It helps to understand what I am trying to get you to do. Here is the course objective statement which you will find when you do your instructor evaluation at the end of the term:

> This course aims to help students develop creative problem solving skills for the type of problems that come up in theoretical computer science. It also aims to teach a few central ideas in the theory of computation, including formal models, reductions, and undecidability.

Note that the first objective doesn't even mention course material—it says "creative problem-solving skills." Here 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 you keep things in 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 behind you will probably find it impossible to get back on track. I have started using the pop quizzes as a tool to "force" student to stay with the class. The right time to understand a lecture is *during* the lecture or, if you didn't, later that day.

Learn to read (and write) the way a mathematician reads (and writes). Each sentence means something precise. And there's nothing extra. Don't read by trying to get some holistic sense of what is being said; clearly understand the content of each sentence.

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. I mentioned that before.

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 a little less abstract. 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 of these classes, I usually recommend it be ECS 122A.

My door is open

I have learned from student surveys that some students are intimidated by me. They are afraid to see me in office hours. Maybe I will be hungry and eat them for dinner. Please don't be afraid. I am harmless. I love my students.

If you are having personal or academic problems feel free to come and talk. I am always too busy, it is true, but it does not mean that I am disinterested in you, or I won't be able to make time to talk to you.

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 fun!?