Introduction

Margaret M. Fleck

20 January 2009

This lecture does a brief introduction to the course and reviews administrative matters.

1 Introductions

This course is Computer Science 173, Discrete Structures.

Introduce myself, Eric, and any other teaching staff who happen to have come.

2 What is the course about?

CS 173 teaches two different sorts of things, woven together. On the one hand, it provides a survey of basic mathematical objects and techniques, useful in later CS courses. These include propositional and predicate logic, sets, functions, relations, modular arithmetic, counting and probability, graphs, trees, and partial orders. 173 also teaches you how to read and write mathematical proofs.

Formal mathematics is relevant to computer science in several ways. First, it is used to create theoretical designs for algorithms and prove that they work correctly. This is especially important for methods that are used frequently and/or in applications where we don't want failures (aircraft design, Pentagon security, ecommerce). Only some people do these designs, but many people use them. The users need to be able to read and understand how the designs work.

Second, the skills you learn in doing formal mathematics correspond closely to those required to design and debug programs. Both require keeping track of what types your variables are. Both use inductive and/or recursive design. And both require careful proofreading skills. So what you learn in this class will also help you succeed in practical programming courses.

3 Everyone can do proofs

You probably think about proofs as something created by brilliant young theoreticians. Some of you **are** brilliant young theoreticians and you'll think this stuff is fun because it's what you naturally like to do. However, many of you are future software and hardware engineers. Some of you may never think of formal mathematics as "fun." That's ok. We understand. We're hoping to give you a sense of why it's pretty and useful, and enough fluency with it to communicate with the theoretical side of the field.

Many people think that proofs involve being incredibly clever. That is true for some steps in some proofs. That's where it helps to actually be a brilliant young theoretician. But many proofs are very routine. And many steps in clever proofs are routine. So there's quite a lot of proofs that all of us can do. And we can all read, understand, and appreciate the clever bits that we didn't think up ourselves.

In other words, don't be afraid. You can do proofs at the level required for this course, and future CS theory courses. This class will teach you how to build proofs, starting with those that have very simple structure and working up to more complex ones.

4 Do you belong in this course?

The prerequisite are high school math and some programming experience (not necessarily enough to proficiency out of CS 125). You should be fluent with high-school algebra and have taken geometry and the precalculus (logs and trig) course. In other words, your score on the math placement test should have been sufficient to take Math 220 or 221. If you feel shaky about your mathematical fluency, it sometimes helps to take calculus or other technical courses (e.g. physics) first. It's definitely better to take CS 125 first if you have a choice.

You may wish to take the proficiency exam if you have a lot of experience writing proofs. Can you write a proof by contradiction? an inductive proof?

Do you know what it means for a function to be one-to-one, or for a relation to be an equivalence relation? If you want to take it, email your class schedule to Margaret Fleck (mfleck@cs.uiuc.edu) so I can set up a time.

Also email Margaret if you wish to take the honors add-on (CS 196).

5 Getting announcements

After today, routine things like homeworks, reading assignments are put on the web page (bring it up) and announced via the class newsgroup. Only the most critical matters (e.g. exams) are announced in lecture. So, read the newsgroup and watch key web pages on a regular basis.

Discussion sections will start next week. Office hours will be posted soon and will start up next week.

Homework 0 will be posted this Friday, due next Friday. It will cover material that should be familiar to everyone, plus some fairly elementary material that some of you may not have seen, which we'll cover on Friday. And perhaps a few questions that you may find interesting to attack but which will be less familiar. Feel free to look up (books, wikipedia, friends, Appendix 2 of the textbook) any things you have forgotten.

Homeworks are officially due at 11 (start of the second lecture). What this means in practice is that we will bring the collection envelopes to both lectures and they will disappear when we leave the second lecture (around 11:50). But (an important but) you are expected to go to your assigned lecture.

If there is some reason you can't make it to lecture (hopefully a compelling reason), push your homework under the door of Margaret's office before 11:50.

6 Meet the textbook

The textbook is *Discrete Mathematics and its Applications* by Kenneth Rosen, 6th edition. (Hold it up.) Notice that the earlier (5th) edition, international edition, used copies should be ok. Read, or at least skim, section 1.1 of the text for Wednesday.

The textbook is clear and it makes a nice comprehensive reference. In a way, this is good. But it does mean that some of it can seem intimidating at first. We won't expect you to memorize everything in section 1.1: watch what we actually cover in lectures and homeworks. We won't read all of chapter 1 in the first couple weeks. Rather, we'll read some of it now and keep coming back to do bits of it as the term progresses.

7 Administrivia

Walk through the web pages highlighting key policy matters, e.g. cheating policy, late homework policy, how course averages are computed.