15-251 | Great Theoretical Ideas in Computer Science

Syllabus

Welcome to 15-251! This course will take a philosophical and historical perspective on the development of theoretical computer science. From using a pile of stones to represent and manipulate numbers, humans have progressively developed an abstract vocabulary with which to mathematically represent their world. The ancients, especially the Greeks, realized that they could consistently reason about their representations in a step-by-step manner. In other words, by computing in abstract models, they could describe and predict patterns in the world around them.

Starting with ancient algorithms for arithmetic, we will revisit the development of mathematics from a computational point of view. Conversely, we will mathematically study the nature of computation itself. What is computation? What is computable, in principle? What is especially easy, or especially hard to compute? To what extent does the inherent nature of computation shape how we learn and think about the world?

Prerequisites: *15-100/111* and *21-127*.

Organization

Lectures will be held every Tuesday and Thursday at 3:00-4:20. We strongly recommend that you attend every lecture; there will be a short quiz every two weeks as a sanity check and class participation will factor into your final grade.

In addition, weekly recitation sections will be held on Mondays. Recitations will be used to supplement lecture material and practice working on problems in small groups. Attendance is MANDATORY.

Resources

The course website is located at *http://www.cs.cmu.edu/~15251*. Everything you could possibly want to know about the course is located there.

References. Historically, this course has not had a physical textbook: we cover a diverse amount of material and no single book contains it all. However, you'll notice that part of the course website is a wiki. We'll post basic notes for most of the lectures online, but feel free to modify or improve on them (indeed, you are strongly encouraged to do so).

Getting Help. If you have a question about a lecture concept or wording on a homework problem, there's a good chance that other students in the class have the same question. Thus, we recommend posting to the class discussion board: *cyrus.academic.cs.15-251*. *Please keep discussion polite and be careful not to give out information about the homework solutions.* If you have a more specific question, we recommend emailing your TA (contact information is located on the website.)

Additionally, everyone on the course staff will have weekly office hours - times and locations are posted on the course website.

Grading

Your grade will depend on the following factors.

Homework. There will be twelve weekly homework assignments, which may occasionally include some programming. They count for **35%** of your final grade and we will drop your lowest homework grade. More information about homework is located below.

Exams and Final. There will be three exams (**30%** of your grade), and a final (**30%** of your grade). Your lowest exam grade will only count for **6%**, and the other two will count for **12%** each

Quizzes. There will be a quiz every other Thursday in lecture as a sanity check. In total, these count for **5%** of your grade. We will drop your lowest quiz grade.

Homework

Homework is the heart and soul of this course. Solving the problems is the only way to gain mastery of the material. Plus, putting the effort in now will alleviate suffering when you get to higher-level theory courses :).

Typesetting. *You must typeset your answers to the homework sets*. Almost all technical and scientific publications are produced using a typesetting system called LaTeX and we strongly recommend it: once you get past the initial learning curve, it's the most painless way to type up your homework. More information on LaTeX is available on the wiki. However, any software that can typeset equations is fine (e.g. Microsoft Word, Adobe Framemaker.) *Writing your work by hand (and scanning it or taking pictures of it) is not acceptable.*

Submission. Homeworks should be submitted electronically. *We will only accept files in PDF format*. To submit homework *N*, copy your homework file to the following directory:

- */afs/andrew/scs/cs/15251/student/assignment**N**/handin/**userid***

Please see the wiki for more information on converting files to PDF.

Late Work. The good news is that you can submit a homework assignment up to **3 (three)** days late. The bad news is that you will lose **10** points for each day it is late. *Late work makes a class much harder to administer. It also hurts you. Please try to avoid it.*

Programming assignments may be resubmitted any number of times throughout the 3-day grading period. Each night at midnight, starting on the night the assignment is due, our scripts will automatically collect all the new handins. We will grade the last submission, taking into account the ten-point-per-day lateness penalty.

If you have a good excuse (such as being amazingly sick), you should contact the professors. For compelling reasons (that extend beyond the fact that you have a lot of work lately and didn't plan ahead), we will excuse you from the lateness penalty.

Collaboration. Discussing ideas and problems with others is an excellent way of learning, and we encourage you to work together in **small** groups. (Please restrict the size of the groups to no more than 3 people, we find that having large groups tends to reduce their effectiveness hugely.) When working on homework problems, however, you need to solve and **write up the actual solutions alone**. It's acceptable to discuss possible approaches with others, but *you should fill in details and write up your solutions independently*. **At the top of your homework sheet, please list all the people with whom you discussed any problem.** Crediting help from other classmates will not take away any credit from you, and will prevent us from assuming cheating if your ideas look similar to theirs. Assigning proper credit is the required practice in all of academia.

Collaboration not only helps get the job done, it teaches you how to explain your (inchoate) ideas to others. This is why we permit discussion of the problems between students. Be careful not to let other people do all the work. If you misuse the opportunity for collaboration in this manner, you will fail the exams and do poorly in the course.

For clarity, here is a partial list of things that would be considered cheating rather than collaboration:

- Showing a draft of a written solution to another student.
- Showing your code to another student.
- Getting help from someone whom you do not acknowledge on your solution.
- Copying a program from someone else.
- Looking at someone else's work on AFS, *even if the file permissions allow it*.

Cheating

Cheating is bad. In fact, we have a separate sheet entirely devoted to this, which you are required to sign and return. All of you are fairly intelligent people and should know what is acceptable and what is not. There are two special points that we'd like to make here, though.

- *Googling for solutions is cheating*. Search engines have become much more prevalent and comprehensive since we started teaching this course, and it's sometimes easy to type related keywords into a search box. If there's a concept you aren't sure of or something you don't understand, *ask us or email us*. We can help you or point you to information that's guaranteed to be high quality.
- Please do not look up previous years' solutions or consult with people who have taken this course before. Some course material will be the same as in previous years. This is not because we are lazy. It takes years to develop good problems. The only reason to change them is to make cheating more difficult. It is far better for you to work on the most excellent problems that we have been able to find in over a decade of teaching. We appeal to your sense of honor because this is what is optimal from a pedagogical point of view.