## CMSC 151: Introduction to Computer Science 1

The University of Chicago, Fall 2011

`http://www.classes.cs.uchicago.edu/archive/2011/fall/15100-1`

**Welcome!** The overarching goal of CS 151 is to introduce students to the subject of computer science by means of an introduction to computer programming. (Computer science and computer programming are not the same thing.)

The more specific course goals are these:

- to design data structures to solve specific computational problems,
- to process those data structures in several ways, most importantly by means of structural recursion,
- to learn to recognize and exploit common computational patterns, and
- to analyze the efficiency of computer programs.

In striving toward these goals, students will become acquainted with a selection of classic data structures and algorithms. We use the Racket programming language as our platform. By the end of the course, students will know how to use computer programming as a flexible, reliable, efficient and comprehensive method for analytical problem solving. Furthermore, students will discover, in future work, that the experience gained in this course applies to programming in any language.

**Instructor** Adam Shaw, email: `ams@`[1], office: Ryerson 157.

**Teaching Assistants** Erik Bodzsar, Sneha Popley (labs); Chris Bun, Negar Mirsattari, Nick Seltzer, Nedelina Teneva (homework).

See the course website for your TAs' office hours.

Rather than contacting the TAs with questions by email, please use the collective question-and-answer system at `http://piazza.com`.

[1] `cs.uchicago.edu`

**Lectures** All lectures are in Ryerson 251. There are two identical sections.

- Section 1: MWF 9:30–10:20.
- Section 2: MWF 11:30–12:20.

The first meeting is on Monday, September 26, 2011; the last meeting is on Wednesday, November 30.

**Coursework** Coursework is partitioned into labs, homework assignments, and lab exercises.

**Homework** There will be homework assignments, roughly weekly. Some of them will be longer project-style assignments spanning several weeks.

**Labs** Students must register for and attend lab each week. There are six lab sections, as follows:

> W 12:00–1:20; W 3:00–4:20; W 4:30–5:50; Th 12:00–1:20; Th 3:00–4:20; Th 4:30–5:50.

There will be eight labs, since Thanksgiving is during the ninth week, and the tenth week ends on a Wednesday. Week by week, lab sessions are the same as one another, but you must attend the same session each week to maintain balanced numbers. All labs are in JRL A01C (the MacLab).

**Exams** There will be a 50-minute midterm exam in class, and a final exam.

The final grade will be computed according to the following formula: homework 35%, labs 25%, midterm exam 15%, final exam 25%. I will grade on a curve, so what constitutes an A, B, *etc.* will be determined by the best marks.

**Late Policy** Late work will not be counted, with the following exception. You have one 24-hour extension on any lab or homework assignment, no questions asked. We will keep track of who has and has not used their extension. To use your extension for a particular piece of work, you must contact the instructor *in advance* of its deadline.

(We will also accept late work in the case of extraordinary circumstances, such as family emergencies. Having a lot of other work to do is not an extraordinary circumstance.)

**Text** *How to Design Programs*, Felleisen *et al.*, ISBN 0-262-06218-6. The textbook is available on campus at the Seminary Co-op Bookstore[2]; you can of course find new and used copies at your favorite online bookstore too. The full text is also available online at `http://www.htdp.org`.

**Software** *DrRacket*, available at `http://racket-lang.org`, and *subversion* (details TBA).

**Schedule of Topics and Readings** (subject to change)

| **Lectures** | **Topic** | **Readings** |
|---|---|---|
| 1 | numbers and expressions | chapters 1–2 |
| 2–4 | functions | 2–6 |
| 5–6 | structures, unions | 7–8 |
| 7–8 | lists, structural recursion | 9–10 |
| 9–10 | trees | 14, 16 |
| 11–12 | searching, sorting | 12, 14 |
| 13–16 | functions on multiple variants, local definitions | 17, 18 |
| 17–20 | mutual recursion | 15, 19–21 |
| 21–24 | higher-order programming | 22, 24 |
| 25–28 | generative recursion, accumulation, graphs | 25–32 |

**Honesty** In this course, you must, as in every course, adhere to college-wide honesty guidelines as set forth at `http://college.uchicago.edu/policies-regulations/academic-integrity-student-conduct`. The college's rules have the final say in all cases. My own paraphrase is as follows:

- Credit your sources.
- Document all collaboration, no matter how small.
- Do not copy the work of anyone else.
- Do not allow anyone else to copy your work.

If you break any of these rules, you will face serious consequences. If you ever have any questions or concerns about honesty issues, raise them with your instructor, early.

**Enjoy yourselves!** There is a lot to enjoy in this introductory course. To get the most out of it, start your work well ahead of time and seek help when you are stuck. The course is meant to be challenging, but, beyond a certain point, it is not profitable to be stumped. We have lots of ways to get you going again if you find yourself unable to make progress. Avail yourselves of office hours and online support.

[2] 5757 S. University Ave., `http://www.semcoop.com`.