CMSC 15200: Introduction to Computer Science-2 **Summer 2016**

https://www.classes.cs.uchicago.edu/archive/2016/summer/15200-1/

Lecturer: Connor Imes *Scribes:*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

1.1 Course Syllabus

Welcome! In CS152, we extend our introduction to major computer science topics through instruction in imperative computer programming and various analytical techniques. The specific goals of the course are:

- To comprehend the basic principles of algorithmic problem solving.
- To use basic control constructs and data types to solve problems.
- To learn to refine and improve programs by an iterative process, using the C programming language, including identifying and carefully fixing coding errors.
- To develop an initial notion of computational efficiency.
- To instill programming best practices like using source control (Subversion) and writing code with consistent formatting and good documentation.

Instructor: Connor Imes

Office Hours: Tuesdays 1-3p in CSIL 3.

Teaching Assistants: Yuxiao "Oliver" Zou, Andoni Garcia, and Aidan Sadowski (available via Piazza).

Piazza: As the course gets going, the flow of questions and answers becomes steady. We will use the piazza system to help manage the traffic. You will need a piazza account to participate. Join the course Piazza page at http://piazza.com/uchicago/summer2016/cmsc15200.

Lectures: MWF 1:30-3:30p in Ryerson 251 from July 25, 2016 – August 26, 2016.

Labs (Mandatory!): Wednesdays 4-6p in Computer Science Instructional Lab (CSIL) 3-4, located in John Crerar Library.

Textbooks: There are two recommended texts for the course. Readings are assigned from the first.

- C Programming, A Modern Approach, by K.N. King.
- The C Programming Language by Brian Kernighan and Dennis Ritchie.

These books are available for purchase from the Seminary Co-op Bookstore. Your are not strictly required to purchase the books, as the lecture notes should be enough to get you past the homework and the exam. However, we strongly recommend that you buy the books if you intend to continue working with C, as it can be an invaluable reference (plus the source of many interesting exercises during the course itself). **Sometimes books are still better than the internet!**

Software: All software used in this course is free. Common text editors include (but are not limited to) Emacs, Vim, and Sublime. We use the GNU C Compiler (gcc) for compiling source code. Subversion is our version control system used for submitting labs and assignments.

Evaluation: Students are evaluated for their performance on labs, homework assignments, weekly quizzes, and a final exam. The breakdown is:

- Labs: 10%
- Homework: 40%
- Quizzes: 20%
- Final Exam: 30%

Grades will be scaled at the end of the quarter.

Final Exam: The final will be held during the regularly scheduled class period on Friday, August 26 in the regular classroom.

Late Work: Late work will not be accepted. We are on a compressed schedule, so you must keep up with the material. Exceptions will be granted only in extraordinary circumstances.

Academic Honesty: In this course, as in all your courses, you must adhere to honesty guidelines as set forth at http://college.uchicago.edu/advising/academic-integrity-student-conduct. The University's rules have the final say in all cases. Our own paraphrase is as follows:

- Never copy work from any other source and submit it as your own.
- Never allow your work to be copied.
- Never submit work identical to another student's.
- Document all collaboration.
- Cite your sources.

Please note that sharing your work publicly (such as posting it to the web) definitely breaks the second rule. With respect to the third rule, you may discuss the general strategy of how to solve a particular problem with another student (in which case, you must document it per the fourth rule), but you may not share your work directly, and when it comes time to sit down and start typing, you must do the work by yourself. If you ever have any questions or concerns about honesty issues, raise them with your instructor, early.

Advice: Writing code that does what it is supposed to do can be joyful, even exhilarating. By contrast, fighting for hours with broken code is misery. We would like you to help you experience more of the former and less of the latter. Work methodically. Start your work well ahead of time. Beyond a certain point, it is not profitable to be stumped. If you have made no progress in some non-trivial chunk of time, say, one hour, it is time to stop and change your approach. Use one of our many support mechanisms to get some assistance. We will help you get going again when you are stuck.

Acknowledgements: A large amount of the material I use in this course was generously supplied to me from previous instructors (and in particular Lamont Samuels, Adam Shaw, and Matthew Rocklin). Thank you for graciously helping me with this course.