EE 200: Accelerated programming for graduate students (a.k.a. "Machine-centric approach prog.")

Fall 2020
Online

Welcome to EE 200! This course is a fast-paced introduction to programming in C and C++, as well as data structures and algorithms analysis. The aim is for you to become a competent programmer, not merely for you to learn C++ syntax. As a result, we will spend parts of the course discussing more general themes in programming languages and software development, and we will use a set of development tools widely employed in embedded systems contexts.

After successfully completing this course, you will be able to:

- Read, write, and debug code written in C and C++, using all of the core features of both languages.
- Descibe the central ideas of object-oriented design, and write software using object-oriented principles.
- Describe the interface for abstract data structures such as vectors, trees, stacks, queues, and graphs, and give example implementations for these.
- Compare algorithms using Big-O analysis.
- Use standard Unix/Linux development software tools, including SSH, Vim or Emacs, GDB, Git, Make and Valgrind.

Communication:

Steven Bell `sbell@ece.tufts.edu`

Halligan 228D

Office hours will be hosted on Zoom, and will be scheduled based on the availability of the class.

To minimize distraction, I generally only check email a few times a day. However, I will make a strong effort to answer all messages within 24 hours on weekdays.

All materials will be posted on the course website: `http://www.ece.tufts.edu/ee/200/`

If you have a general question about the course content or course logistics, please post on Campuswire rather than emailing me. That way anyone can answer the question, and everyone benefits from the response. You should have received the sign-up link via email.

Prerequisites:

There are no prerequisites for the course except for graduate standing. In particular, we will not assume any programming experience in any language.

Textbook:

We will use the textbook "All of Programming" by Andrew Hilton and Anne Bracy. Information about the book is online at `http://aop.cs.cornell.edu/index.html`. The text is only available as an e-book, which you can download from Google Play for the bargain price of $10. Once you've bought the book, it's possible to download it as a PDF or transfer it to various e-readers.

Class sessions:

There are no scheduled class sessions for this course.

Programming is a bit like playing a sport: you improve by practicing, both on your own and under the direction of a coach. You don't improve much by watching professionals or by

listening to hour-long lectures about your technique. As a result, this course is structured to let you *practice* programming, rather than listen to lectures about programming.

The content is broken up into a series of small modules, and we will cover 1 or 2 modules per week. For each module you will do the following:

1. Read the assigned sections in *All of Programming*, and complete a short quiz with questions from the reading. Each quiz will also contain a space for you to ask questions or identify material that was particularly confusing. I will use these responses to make additional videos or to help in office hours.
2. Watch the videos. For each module, I will post a few videos to clarify difficult topics from the textbook, show examples of code, or demonstrate relevant tools.
3. Complete the programming assignments. These will usually be due a few days after the corresponding quiz is due.

Assignments:

Each assignment has two grading components: *correctness* and *style*. Correctness will be assessed with a series of automated tests which execute your function or program and ensure that it produces the correct result. Style will be manually graded, and includes your algorithm design, coding style (variable naming, indentation, braces, etc.), and documentation (comments and `README`s).

Because everything in this course builds on previous material, it is critical to have a solid understanding of each concept or technique before moving on. To encourage this, you will have a chance to revise and resubmit each assignment and get up to 75% credit for the correctness points you missed.

Late work:

Work submitted after the deadline but before we release the autograder results will be accepted for 90% credit. Work submitted after the autograder results are released will count for partial credit as if the original submission was a zero (i.e., it will count for 75% credit).

Exams:

There will be two exams, one roughly halfway through and one at the end of the course. These will be structured like online coding interviews: you'll talk with me on Zoom for about half an hour, I'll ask you various questions about programming, and you'll type code in a Google doc. More details will be provided as we approach the first exam.

The course content is cumulative, and so the exams will be as well. However, the final exam will be weighted toward the material in the second half of the course.

Grading:

Grades will be assigned on an absolute scale (not curved), with the following components:

Pre-class quizzes: 10%

Homework: 50%

Exams: 40%

Plagarism:

Collaboration is essential in real-world software development, so you are welcome (and encouraged) to discuss approaches and solutions with other students, and to help each other debug code. Likewise, is is expected that you will use online resources such as cplusplus.com and Stack Overflow as you develop and debug your code.

However, as with learning a sport, the person who puts in the effort gets the gains, and therefore it is essential that every student put in their own effort to learn. Concretely, this means that each student must write all of their own code. It is not acceptable to copy code from another student or from the Internet, whether your are copy-pasting or just typing from another screen.

In your submission, you must document anyone you worked with and any sources you used to develop your code (which is good practice in any software development context). Homework submissions will be run through MOSS, a tool widely used in academia to detect plagarism.

Schedule:

The schedule of class topics and assigned readings is on the course website. It may be adjusted slightly to meet the needs and pace of the class.