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

Fall 2024
T/Th 4:30pm, JCC 076

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 112 (click here for directions)

Office hours:

- Mondays 3pm-4:30pm
- Thursdays 10am-11:30pm
- I’m also available other times by appointment (generally mornings Monday-Thursday, as all of my classes are scheduled for the afternoons).

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 Piazza rather than emailing me. That way anyone can answer the question, and everyone benefits from the response. You will receive 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 will be available for $20 through the online discussion platform Perusall. You will receive a link to join Perusall at the beginning of the course.

Class sessions:

Class sessions are on Tuesdays and Thursdays, 4:30-5:45pm in Bromfield-Pearson 003.

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.

Before class: You will read the assigned sections in *All of Programming*, complete the quiz, and join the discussion about the reading on Perusall. I will use your questions, comments, and responses to focus our time in class.

In class: We will begin with a lecture on the topics that came up from the reading, and on any relevant tools or examples that were not covered by the book. We'll pause frequently for group discussion, practice problems, and Q&A.

The rest of the class time will be devoted to getting started on the day's homework assignments and working through any setup and configuration issues or basic questions about the assignment. It will also be an opportunity for me to talk through feedback on your code and answer any questions you have about previous assignments.

After class: You will work to complete the assignments, and submit them by the following week.

Assignments:

Coding assignments will be assigned nearly every class session. In general, they will be due a week after they are assigned.

Each assignment has three grading components:

- **Initial correctness**: An autograder will run your code against a number of test cases, and will flag any errors. You will be able to see this score shortly after the submission deadline.
- **Resubmission correctness**: After the autograder results are released, you will have an opportunity to resubmit your code and earn up to 75% for the points you initially missed.
- **Code merged**: After completing each assignment, you will submit a "pull request". As in a profesional software project, I will review your code and will ask you to make some changes before I accept your work. I will be looking at your algorithm design, coding style (variable naming, indentation, braces, etc.), and documentation (comments and `README`s).

Late work:

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).

If you have extenuating circumstances (major illness, family emergency, etc) such that you will need to turn in multiple assignments late, please reach out to me *before the deadline* and we can work out an appropriate schedule.

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 video chat with me for about half an hour, I'll ask you various questions about programming, and you'll type code in an online coding environment. 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 / Perusall engagement: 10%

Homework: 50%

Exams: 40%

Academic integrity:

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.

Likewise, while AI tools such as Github Copilot and ChatGPT have promise as development tools which amplify the ability of skilled developers, the emphasis in this course is on building foundational skills with C and C++. *It is not acceptable to use an AI tool to generate code for any assignment related to this course, even as a reference or starting point.* We may experiment with using ChatGPT to give feedback on error messages or to debug broken code; if we do, we will discuss acceptable uses of AI tools in more detail in class.

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 may 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.