CS107 Course Information

Instructor: Jerry Cain

Office hours: Mondays, 9:00 – 10:30 a.m. in Gates 192
Wednesdays, 9:00 – 10:30 a.m. in Gates 192

Lectures: MWF 11:00 – 11:50 a.m.
Gates B01

Units: 5 units. Only matriculated graduate students can register for fewer than five units. All undergraduates need to enroll for all five.

We already have 239 (!) students signed up (as of last night) and that number will probably climb. I'll be working to hire additional TAs, so stay tuned for more names and email addresses. CS107 TAs attend lectures, hold office hours, grade your program submissions (within one week of submission, or I cane them), and help me grade the midterm and final exams. The feature here is that all of them have taken CS107 with me before. They know the material inside and out, and because they've written all of the very same programs you'll be writing, they already

know what your questions will be. Be happy they are here, because they make CS107 that much better of a course.

Sections: We're currently having a really hard time scheduling a discussion section, but rest assured there will be one, probably on Tuesdays. We'll post an announcement on the web site as soon as we converge on a section time.

Section attendance isn't required any more than lecture attendance is, but you can only benefit by going. We'll begin next Tuesday and spend the hour covering the various development tools you'll be using to write code. If you attend just one section all quarter, this should be the one. Those of you with little or no experience in Unix will definitely want to go. The rest of the discussion sections will follow a more traditional format, where we answer student questions and work through prepared problems relevant to lecture material. In all cases, you are fully responsible for anything that appears in a section handout, even if you don't go or watch online.

Prereqs: The prerequisite for the class is programming and problem solving at the CS106B/X level. CS106B and CS106X started teaching C++ about four years ago, so I'm assuming that virtually all of you know a reasonable amount of C++. If you don't know C++, then you have your work cut out for you these first few weeks. I'm happy to give a crash course in C++ if I sense a demand, but it's ultimately your responsibility to teach yourself enough to brave the first two programming assignments.

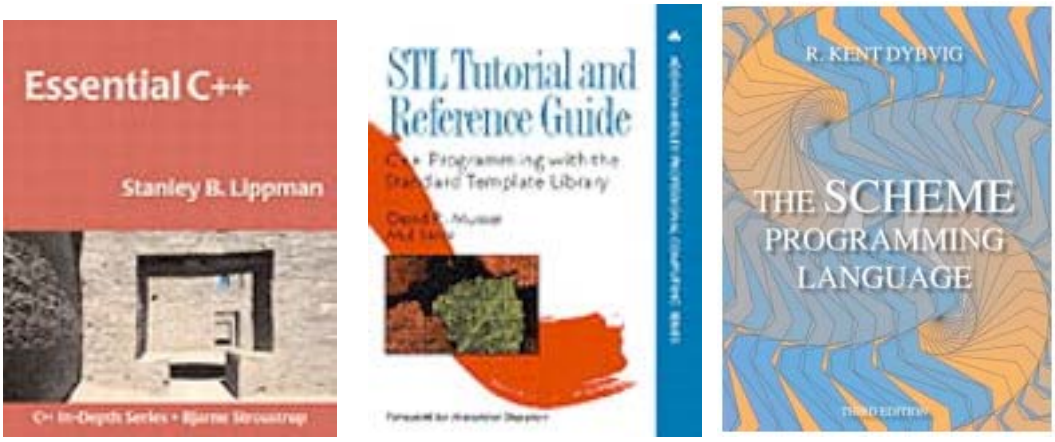
The hard prerequisite: you should be comfortable with arrays, pointers, references, classes, methods, dynamic memory allocation, recursion, linked lists, binary search trees, hashing, iterators, and function pointers. You should be able to write well-decomposed, easy-to-understand code, and do so not just because we make you, but because you understand the value that comes with good variable names, short function and method implementations, and thoughtful, articulate comments.

Readings: Virtually all course material comes in handout form. What little doesn't get typed out comes up during lecture. But in general, you're responsible for all lecture material, including material not covered in a handout. Those of you eager for additional reading should inspect the course web site, where I present a list of relevant texts, publications and online articles. I'm particularly fond of two C++ books and one Scheme book:

1. Stanley B. Lippman, Essential C++.
2. David R. Musser, Atul Saini, STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library.

3. R. Kent Dybvig, The Scheme Programming Language: Third Edition.

The three books look something like this:



The first two are advanced textbooks, but you're advanced C++ programmers as of today, and you deserve a more sophisticated treatment of the material. Those of you willing to do some independent reading every once in a while will come to appreciate them. The third is a handy reference for learning the Scheme dialect of LISP, and those of you who end up digging Scheme will like this book.

Software: The UNIX workstations at Terman and in the basement of Gates provide all of the development tools necessary to program in the various languages we'll be studying. In most cases, you'll be able to work on any platform you choose, provided you port your code back to the workstations for final testing and submission. But honestly, it makes more sense for you to work directly under UNIX and not bother with any porting efforts during the 11th hour. Virtually all systems classes post-CS107—CS140, CS143, CS244A, etc.—require UNIX, so it makes sense to get the experience now when you have a huge staff willing to help you get used to it.(

SUNet Ids: Everyone needs a SUNet ID in order to log into Stanford's UNIX machines. Returning students have most certainly taken care of this by now, but some new SCPD students may not have.

Class email: There is a class mailing list that will be used for important or late-breaking announcements. All students enrolled in CS107 are automatically subscribed to the mailing list. The list server is in touch with Axess and automatically includes everyone enrolled in the course. **Please** make it a point to register for CS107 sometime before the weekend. You should also inspect your `stanfordyou.stanford.edu` privacy settings to ensure that this email address of yours is viewable by the Stanford community. If not, then your email address is excluded from the mailing list, and you'll miss out on all the gossip.

Web 2.0 For the savvy Web 2.0 fans, you can follow cs107 at `http://www.twitter.com`. And I've even set up a Facebook group called CS107: Programming Paradigms, which you can join by visiting `http://www.facebook.com/group.php?gid=13800096012` (provided you have a Facebook account, of course.)

Staff email: You're always welcome to email me directly, but I occasionally escape and ignore my email for several hours at a time. Unless you need to address me specifically, you should probably send mail to All of us poll regularly enough that you can expect a response within a few hours during the workweek, and within 24 hours on the weekends. Very often we get back to you within a few minutes.

Newsgroup: Our class newsgroup is `su.class.cs107`. In the past, we've blown off the newsgroup, but I'm now marketing it as the best way for you to ask each other questions. Unlike , anyone can respond to any posting. I'm hoping to build a little social network where all the CS107 students can help each other out when deadlines are fast approaching and TAs don't seem to be around.

Grading: This class is offered with both the graded and **CR**/**NC** options. The course grading is divided between programming assignments, one midterm, and a final exam. The approximate grade breakdown is:

Programs	40 %
Midterm	25 %
Final	35 %

To receive a passing grade, you must pass at least one of the two exams. Restated, if you fail both the midterm and the final exam, then you will fail the class in spite of any astounding performances on my programming assignments. Grades in CS107 tend to be very good, where between one third and one-half of the students pull some form of an A, and all but a handful end up with a B or better.

Exams: The (open-notes, open-lecture-notes, closed-computer) midterm will be given on Wednesday, May 7th from 7:00 p.m. – 10:00 p.m. in a location to be announced. My midterms aren't intentionally written to take three hours, but in the interest of removing time pressure I give what I hope is more than an adequate amount of time. If this time doesn't work for you, then you'll need to take the exam during some three-hour window earlier in the day. Because the exam is held outside the normal class time, I am more than happy to accommodate all reasonable requests.

The three-hour, open-book, open-lecture-notes, closed-computer final exam will be offered twice during finals week.

Monday, June 9th at 8:30 a.m. **and**
Monday, June 9th at 3:30 p.m.

The first is the normally scheduled time, and the second is a time slot later that day where I suspect everyone will be free. Just pick the time that's more convenient for you, and show up. I have near zero flexibility when it comes to dealing with final exam special cases, so you'll need to figure out how to make one of these two times. Oh, the final exam will emphasize material not tested on the midterm, although it is technically cumulative and everything is fair game.

Late Policy: The class material builds on itself and getting behind is taboo, since it tends to impede progress on the following assignments. Late submissions are frowned upon big time, so any assignment turned in late will be assessed a penalty of 10% per day (24-hour period). Assignments will not be accepted more than **five** days after the original assignment

due date. The TAs are as busy as you are, and it's unfair to have them go back and grade a second wave of late assignments when I'm pressuring them to grade the on-time ones as quickly as possible.

All that being said, we all have our emergencies. I have my own little breakdowns from time to time, and I expect you do too. Instead of demanding that you ask for special allowances on an individual basis, I will give each of you the privilege of granting yourself small extensions whenever crises arise. You get **five** free "late days" (24-hour periods) that you may use to extend the due dates of any assignment without penalty. You may use one, two, or even all five of your free late days for any particular assignment (except the last one, which may impose a harder deadline), or you may distribute them across several different assignments. **Understand that requests for additional late days are consistently denied unless the five free ones were used for the very best of reasons**. If you would be comfortable going up in front of the class and explaining why your situation is exceptional and requires special consideration, then you probably have one of the very few legitimate arguments for an additional late day or two. (Once all your free late days are gone, you can still hand work in late, but the 10% penalty kicks in.)

Incompletes: You may take an incomplete in the class if and only if you've completed everything except the last assignment and the final exam. Unless you need the incomplete because of a personal emergency (death of a close friend of family member, severe illness requiring you to remain in bed or in the hospital), I will levy a moderate to severe penalty, depending on when you finish the course.

Honor Code: Although you are encouraged to discuss ideas with others, your programs are to be completed independently and should be original work. Whenever you obtain significant help (from other students, the TAs, students in other classes, etc.) you should credit those who helped you in your program write-up, e.g. "The idea to use `qsort` to alphabetize the names came from a discussion with my TA, Kemble Scott." Any assistance that is not given proper citation will be considered a violation of the Stanford Honor Code.

To be even more specific, you are not allowed to collaborate on the coding of your programs, nor are you allowed to copy programs or even parts of programs from other students. The following three activities are among what I consider to be Honor Code violations in this course:

1. Looking at another student's code.
2. Showing another student your code.
3. Discussing assignments in such detail that you duplicate a portion of someone else's code in your own program.

Unfortunately, the Computer Science Department sees much, much more than its share of plagiarism. Because it's important that all cases of academic dishonesty are identified for the sake of those playing by the rules, we reserve our right to use software tools to compare your submissions against those of all other current and past CS107 students. It isn't my intent to create some Big-Brother-Is-Watching environment with moles and surveillance cameras. I'm just being clear about how far I'll go to make sure the consistently honest feel their honesty is valued and rewarded. If the thought of copying code has never crossed your mind, then you needn't worry, because I've never seen a false accusation. But if you're ever tempted to share code—whether it's because you don't understand the material or you **do** understand but you're short on time—then you need to remember this paragraph is here.