

## CS106X Course Information

---

**Instructor:** Jerry Cain  
**E-Mail:** [jerry@cs.stanford.edu](mailto:jerry@cs.stanford.edu)  
**Cell phone:** (415) 205-2242, between 7:00 a.m. and 9:00 p.m.  
**Office:** Gates 192  
**Office hours:** Mondays and Wednesdays 1:15 p.m. until 4:05 p.m.

Don't take the minimal office hour offerings as a signal that I don't want you to drop by. In practice, I interact with most students over email, and most CS106X students rely on the section leaders for their one-on-one question-and-answer, so scheduling lots of office hours often amounts to me being in my office all by myself for a few hours. If the provided times aren't good for you and you'd like to see me, you can always schedule an appointment, or you can even telephone me at the above cell phone number to chat (or say you want to stop by.)

**Prerequisites:** CS106X is the more advanced of the two courses teaching introductory programming abstractions and algorithms. CS106X is designed as an alternative to the more sensibly paced CS106B, because many students—self-taught hackers, exceptionally strong CS106A students, and AP Java graduates—prefer a more intense treatment in the company of other aficionados.

AP Java and CS106A are all about basic programming practices—expressions, control idioms, decomposition, algorithmic thinking, class design, object orientation, simple inheritance, and client use of arrays and maps. CS106X teaches advanced programming and abstraction techniques, worrying first about C++ language mechanics and eventually focusing on topics such as recursion, C++ vectors, lists and maps, and the implementation techniques used to build custom, dynamic data structures.

**Lectures:** MWF 9:00 – 9:50 a.m.  
Skilling Auditorium

My lectures tend to be (and are intended to be) very conversational, feel-good and informal, working through material at an intense but manageable pace. I go through a good mix of examples—some drawn verbatim from the reader, but most are my own. I often stop mid-topic at 9:50 one lecture and pick up as if I never stopped talking two or three days later at 9:00.

**Labs:** In addition to the three lectures every week, you'll also participate in a 60-minute programming laboratory (beginning the week of October 2<sup>nd</sup>). The

three lectures are optional, in that you don't need to attend them at all if you're able to keep up with the material and not have it impede your ability to get your work done. However, programming laboratory attendance is mandatory, and your presence and willingness to work on the laboratory exercises in good faith contributes to your final course grade. The labs are part discussion section, part coding, and the problems you'll be discussing and coding up will be distributed in paper and PDF format at least two days ahead of time. Those with laptops should bring them to lab, but those without laptops shouldn't worry, as we'll be pairing everyone up for the coding portion and will take care to pair those without laptops with those who have one.

There are several programming lab times to choose from, and those times will be published to <http://cs198.stanford.edu/section> by Thursday, 5:00 p.m. of this week. You'll have between Thursday at 5:00 p.m. and Sunday at 5:00 p.m. to view your options and state your preferences. In the past, we've been able to assign the vast majority of students to their first choices, and virtually all to one of their top two. If after you've been scheduled to a lab time you find that you can't regularly attend, you can contact me if we failed to reasonably accommodate your schedule, or you can just return to the CS198 web site and switch sections if your initial assignment worked out well but for whatever reason things changed.

**Readings:** The class textbook is the course reader *Programming Abstractions in C++* by Eric Roberts (with C++ edits by Julie Zelenski). The course reader should already be available at the Stanford Bookstore.

In addition to the reader, we distribute a good number of handouts, chockfull of additional material and examples. All of the handouts are posted online to the course web site in PDF format, and it's our expectation that you read the handouts online, printing them out yourself if that suits you better. We will provide hardcopies of some handouts—lab handouts, assignments, and practice exams—when it's clear that having a paper copy available is uncontroversially and substantially more convenient for everyone.

**Software:** Programming assignments can be written on either Macintosh or Windows PC computers, using either XCode (on the Macintosh) or Visual Studio C++ (on the PC). More information on these two programming environments will be provided in separate handouts.

**Mailing List:** There is a class mailing list that will be used for important announcements that just can't wait until lecture. All students enrolled in CS106X are automatically subscribed to the **cs106x-aut1112-students@lists** list. The list server is in touch with Axess, so if you're signed up for the course, you're on the mailing list already. Please make it a point to sign up for the

course as soon as possible, as I tend to send a good number of announcements out during the first week or two, and I don't want you to miss out.

**Programs:** There are six or seven programming assignments, and it's possible I'll throw in a written problem set for color. The assignments are serious projects, and they get more and more difficult as we cover the more advanced material. The only way to learn programming is to work at it, so expect to spend lots of time in front of a computer. Your assignments are graded interactively in a one-on-one session with your section leader. In general, section leaders will return your assignments within one week of the day you submit it.

**Exams:** There will be two examinations

First Exam:	Tuesday, October 25 <sup>th</sup>	7:00 p.m. – 10:00 p.m.
Second Exam:	Wednesday, November 30 <sup>th</sup>	7:00 p.m. – 10:00 p.m.

The first exam will cover the first four weeks of the course, and the second exam will cover the first eight weeks of the course, focusing on the material not covered on midterm one. Each exam could be administered as a two-hour exam, but I've scheduled three hours so as to do my reasonable share to remove whatever time pressure might otherwise be present.

There will be no final exam, which would have normally been held on the morning of Wednesday, December 14<sup>th</sup> from 8:30 – 11:30 a.m. Instead, your final assignment will be due at 11:30 a.m., at the time when your final exam would have ended.

**Grading:** Your final grade will be computed as follows:

Programs	50 %
Lab Participation	10 %
First Exam	20 %
Second Exam	20 %

Assignments are graded on a bucket system, as we want to de-emphasize the letter or number grade and would like you to focus more on our general impression and feedback. But in the interest of transparency, here is a rough description of the various buckets and the numbers they correspond to.

- + Given to an exceptionally strong submission that not only meets the requirements, but exceeds them in some significant, algorithmically interesting way. In general, I see less than 2% of assignments getting +'. The + is ultimately recorded as a 100 in the spreadsheet, since it's clearly A+ work.

- √+ Given to a solid submission that gets the job done and contains at most a very small number of trivial errors. In general, 35-40% of assignment submissions get the √+, which maps to a 96.
- √ Given to a good submission that gets most of the job done and contains one or more major errors, or a significant number of minor ones. In general, about 45-50% of assignment submissions get a √, which maps to an 88 come spreadsheet time. This is the most controversial grade, because CS106X students don't like getting B+'s. However, when we give them, it's because the program wasn't as good as it could have been and there were more impressive submissions.
- √- Given to a submission that does most of the work, but contains enough problems that even a √ isn't warranted. In practice, √-'s are rare in CS106X. The √- maps to an 80 come spreadsheet time.

There are other bucket grades, but they are super duper rare enough that I don't need to describe them. You should try to not get them. ☺

For each assignment, we also issue a companion grade summarizing our evaluation of your overall design, style, and code clarity. While issuing grades, we're very open to different approaches, and penalties are imposed only when there are clear arguments that you overcomplicated an issue or your general coding style is sloppy. Style grades are also bucketed, but we only issue √'s, √+'s, and √-'s.

The class median on the first exam tends to be high—typically above 80 percent, while the median on the second exam tends to be between 70 and 80. When an exam median is 80 or above, your raw exam score contributes verbatim to your final average. When the exam median is below an 80, I curve the highest grade to a 100, the median grade to an 80, and everything else is linearly interpolated.

Those with a 90.0+ average (around a third of you, typically) at the end get some form of an A. Those with 80.0+ averages who don't make it to 90.0 (all but a handful of you) typically get some form of a B, and so forth.

### **Fair Access**

Students who may need an academic accommodation based on the impact of a disability must initiate the request with the Student Disability Resource Center (SDRC) located within the Office of Accessible Education (OAE). SDRC staff will evaluate the request with required documentation, recommend reasonable accommodations, and prepare an Accommodation Letter for faculty dated in the current quarter in which the request is being made. Students should contact the SDRC as soon as possible since timely notice is needed to coordinate accommodations. The OAE is located at 563 Salvatierra Walk (phone: 723-1066).

**Late policy:** The pace of this course makes it difficult for students to catch up once they have fallen behind, so I encourage you to submit all of your assignments on time. Of course, we're all busy people, so I understand when you can't meet each and every deadline I put before you.

Here's how I handle lateness:

You get **three** free late days, and you consume one late day any time you hand in work between 1 second and one class period late. You may never hand in an assignment more than two class periods late, as it encumbers your section leader's ability to deliver feedback in a timely manner.

There is one exception: you may not use any late days at all on the final assignment, as that final assignment is due in the middle of final exam week and the section leaders should be able to launch into the grading effort at 11:31 a.m. on Wednesday, December 14<sup>th</sup>.

It is also strongly recommended that you do not take late days on the assignments with due dates immediately preceding exams. In the spirit of an accelerated course, I try to teach as much interesting material as possible during the course of the 10 weeks you're with me, and that's difficult to do if we have to slow down every time as exam approaches.

**Incompletes:** I will only grant incompletes—in the form of NC's and NP's—in the event of a severe illness or a family emergency that makes it impossible for a student to finish his or her work during the regular academic quarter. Unless your circumstances are particularly extenuating, you must clear the incomplete by January 15<sup>th</sup>, 2012.

**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 section leaders, students in other classes) you should acknowledge this in your program write-up, e.g. "The idea to use insertion sort instead of quicksort to alphabetize the list of names was actually my section leader's idea." 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 while physically coding, nor are you allowed to copy programs or parts of programs from other students. The following three activities are among the many considered 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 Honor Code violations. 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 CS106 students. It isn't our intent to create a Big Brother environment with spies and surveillance cameras. We're just being clear about how far we'll go to make sure the consistently honest feel their honesty is valued.

If the thought of copying code has never crossed your mind, then you needn't worry, because I've never seen a false accusation go beyond a conversation. But if you're ever tempted to share code—whether it's because you don't understand the material, or you do understand but just don't have enough time to get the work done—then you need to remember this paragraph is here.