CS106X Course Information

Instructor: Jerry Cain
E-Mail: jerry@cs.stanford.edu
Office: Gates 192
Cell Phone: 415-205-2242
Office hours: MWF 3:00 – 5:00 p.m., and by appointment

Website: http://cs106x.stanford.edu

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 some students—self-taught programmers, 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 basic client use of arrays, lists, and maps. CS106X teaches advanced abstraction techniques, worrying first about C++ language mechanics and eventually focusing on topics such as recursion, inheritance, networking, event-driven programming, C++ lists, sets, and maps, and the implementation techniques used to build custom data structures.

Lectures: MWF 1:30 – 2:50 p.m.
Building 380, Room 380D

My CS106X lectures are feel-good, conversational, and informal, while working through material at an intense pace. We go through a good mix of examples—some drawn verbatim from the reader, but most are my own. I often stop mid-topic at 2:50 one lecture and pick up as if we never stopped talking next lecture at 1:30.

Sections: In addition to the three lectures every week, you'll also participate in a 50-minute discussion sections (beginning the week of January 15th). Sections are mostly whiteboard discussion, sometimes with a little bit of coding, and the problems you'll be discussing will be posted well in advance. Those with laptops should bring them to section, but those without laptops shouldn't worry, as we'll be pairing everyone up for any coding portion.

There are several section times to choose from, and those times will be published to https://cs198.stanford.edu/cs198/auth/section/

by Thursday, January 11th at 5:00 p.m. You'll have between Thursday at 5:00 p.m. and Sunday, January 14th at 5:00 p.m. to weigh 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.

Readings: The class textbook is Programming Abstractions in C++ by Eric Roberts, which should be available at the Stanford Bookstore. If you'd prefer, you can download a PDF of the reader from the course website and read from that.

In addition to the reader, we distribute a good number of handouts and slide decks, chock full 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—discussion section handouts, assignments, and practice exams—when it's clear that having a paper copy available is unambiguously better.

Software: Programming assignments can be written on either Macintosh or Windows PC computers using a development environment called QtCreator. More information about QtCreator will be provided online by this Wednesday's lecture, when your first assignment goes out.

Mailing List: All students enrolled in CS106X are automatically subscribed to the **`cs106x-win1718-students@lists`** mailing list. The list server is in touch with Axess, so if you've signed up for the course, you're probably on the mailing list already. Please make it a point to register for CS106X as soon as possible, since we tend to broadcast a good number of announcements during the first two weeks, and we don't want any of you to miss them.

Assignments: There are six or seven programming assignments, and it's possible we'll throw in a written problem set as well. The assignments are serious projects, and they get more difficult as we cover 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, your section leader will meet with you and return an assignment within one week of the day you submit it.

Exams: There will be two in-class examinations

First Exam:	Friday, February 9th	1:30 p.m. – 2:50 p.m.
Second Exam:	Friday, March 9th	1:30 p.m. – 2:50 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 covered during Weeks 5 through 8.

Final Project: In place of a final exam, you'll define, design, and implement a final project that you define from start to finish. The final project can be anything whatsoever, provided the final application matches the size and complexity of any of your CS106X assignments. Your final project must leverage at least one topic taught in CS106X, and would ideally leverage many of them. Your program can be anything at all, and might be informed by your interest in literature, music, art, history, languages, biology, engineering, or anything else you want to incorporate into a well-formed project.

We'll provide more detail about the final project during Week 3, once we've all settled into the class and I've gotten the chance to meet all of you. My expectation is that we'll work through the majority of the course material and the standard assignments during the first eight weeks of the course, and that'll allow for you to dedicate the rest of the quarter—up until March 19th at 10:00am—to fully implement your project.

Grading: Your final grade will be computed as follows:

Assignments	50 %
First Exam	15 %
Second Exam	15 %
Final Project	20 %

The assignments and the final project are all graded on a bucket system, as we want the discussion to de-emphasize the letter grade and instead focus on what *could* have been done differently and what *absolutely needed* to be done differently.

In the interest of transparency, however, here is a clear 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, we see less than 5% of assignments getting +'s. 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 95.

√ 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 Stanford students don't like getting B+'s on assignments. 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 much of the work, but contains so many obvious problems that even a √ isn't warranted. The √- maps to an 80 come spreadsheet time.

- Given to a submission that clearly fails to solve the assigned problem or problems adequately. CS106X students generally don't get these grades, but if you get one it's because something clearly didn't go well. The - maps to a 70 when we work all of my spreadsheet magic at the end of the quarter.

For each assignment, we also issue a companion style grade evaluating your overall design, decomposition, 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 something or your general coding style is subpar. Style grades are also bucketed, but we only issue √'s, √+'s, and √-'s. Functionality counts twice as much as style.

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 as is to your final average. When the exam median is below an 80, we curve the highest grade to a 100, the median grade to an 80, and everything else is scaled up accordingly.

This is the first time CS106X is requiring you design and build a final project, but I anticipate the vast most of you will receive double √+'s. We'll be working with each of you during the brainstorming, design, and implementation phases to ensure a successful and satisfying result.

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 we encourage you to submit all of your assignments on time. Of course, we're all busy people, so we understand when you can't meet each and every deadline we put before you.

Here's how we handle lateness: You get **two** free late days, and you consume one late day any time you hand in work between one second and one class period after the original deadline. Once you consume your two free late days, you can still hand in late work, but your late days are no longer free. For each additional late day, we subtract 2% from your overall homework average. In general, it's wiser to take an extra late day unless you think you're in √ territory already, in which case it probably isn't worth it.

Note that the final project must be completed by March 19th at 10:00am, without exception. Restated, you can't use any late days—even free ones—for the final project.

Incompletes: We only grant incompletes to those who complete all work due prior to the course withdrawal deadline, and only because of a severe illness or a family emergency. Understand that an incomplete is not a giant reset button you get to press to start over. Rather, it's a courtesy extension on some end-of-quarter deadlines to help mitigate a very poorly timed personal crisis. In general, all work must be completed before spring quarter begins.

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 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." Even if you get help from others, the work you submit should uncontroversially be viewed as original work.

To be even more specific**,** you are not allowed to collaborate while actively coding, nor are you allowed to copy programs or parts of programs from other students. The following four 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, or making your code public so that it's searchable and easily discovered online or elsewhere.
3. Discussing assignments in such detail that you duplicate a portion of someone else's code in your own program.

4. Uploading your code to a public repository (e.g. **`github.com`** or **`bitbucket.com`**) so that others can easily discover it via word of mouth or search engines. If you'd like to upload your code to a private repository, you can do so on **`bitbucket`** or some other hosting service that provides free-of-charge private hosting.

Unfortunately, the CS department sees more than its fair 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 use software tools to compare your submissions against those of all other current and past CS106 students. While we certainly don't want to create some Big Brother environment, we do need to be clear 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 heated 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 these paragraphs are here.