CS106B Course Information

Instructor: Jerry Cain
E-Mail: **jerry@cs.stanford.edu**
Cell phone: (415) 205-2242
Office: Gates 192
Office hours: Mondays, Wednesdays, and Fridays, 10:15 a.m. – noon, and by appointment

Don't take the minimal office hour offerings as a signal that I don't want you to drop by. If the provided times aren't good and you'd like to see me, schedule an appointment, or telephone me at the above cell phone number. I'm more than happy to meet or speak unless I'm under my own deadline.

Lectures: MWF 3:15 – 4:05 p.m.
Building 420, Room 040

Website: **http://cs106b.stanford.edu**
Facebook: **http://www.facebook.com/cs106b**
Twitter: **http://twitter.com/cs106b**

Prerequisites: AP Java, CS106A, or equivalent. 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. CS106B teaches advanced abstraction techniques, worrying first about C++ language mechanics and eventually focusing on topics such as recursion, C++ lists, sets, and maps, and the implementation techniques used to build custom, dynamic data structures.

Labs: In addition to the three lectures every week, you'll also participate in a 50-minute programming laboratory (beginning the week of October 1st). The three weekly lectures are optional, in that you don't need to attend them if you're able to keep up with the material. However, programming laboratory attendance is required, and your presence and willingness to work on the exercises in good faith contributes to your final 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 be sure to pair those owning laptops with those who do not.

There are several programming lab times to choose from, and those times will be published to **`http://cs198.stanford.edu/section`** by Thursday, September 27th at 5:00 p.m. You'll have between Thursday at 5:00 p.m. and Sunday, September 30th 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.

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

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 unambiguously better 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 online by Wednesday, September 26th.

Mailing List: All students enrolled in CS106B are automatically subscribed to the **`cs106b-aut1213-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 CS106B as soon as possible, since I tend to send a good number of announcements out during the first week or two, and I don't want any of you to miss them.

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, your section leader will meet with you and return an assignment within one week of the day you submit it.

Exams: There will be one three-hour midterm and one three-hour final exam.

Midterm:	Tuesday, October 30th	7:00 p.m. – 10:00 p.m.
Final:	Friday, December 14th	12:15 p.m. – 3:15 p.m.

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

Exams will be closed note, closed computer, closed book. (We will not be especially picky about trivial syntax issues.) If you can't make the normally scheduled midterm time, then you can take it any time during earlier in the day. However, there is no alternate final exam, so be sure you can attend.

Grading: Your final grade will be computed as follows:

Programs	50 %
Lab Participation	5 %
First Exam	20 %
Second Exam	25 %

Assignments are graded on a bucket system, as we want to de-emphasize the letter grade and instead focus more on our feedback. But in the interest of transparency, 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, I 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 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 Stanford 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 much of the work, but contains enough problems that even a √ isn't warranted. The √- maps to an 80 come spreadsheet time.

There are other bucket grades, but they are rare enough that I don't need to describe them.

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 an issue or your general coding style is sloppy. 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 final 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 one second and one class period after the original deadline. Once you consume your three free late days, you can still hand in late work, but your late days are no longer free. For each additional late day, I subtract 2% from your overall homework

average. In general, it's wiser to take an extra late day unless you already think you're in √ territory, in which case it's probably not worth it. Of course, you should still work to complete the assignment and/or figure out what's preventing it from working.

And, 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.

Incompletes: I 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 poorly timed personal crisis. In general, all work must be completed before winter 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 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, 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.

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 very 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.