Course Information

Instructor Keith Schwarz (htiek@cs.stanford.edu)

TAs Narant Bayanbat (naranb@stanford.edu)
Jinchao Ye (jcye@stanford.edu)

Email The course staff can be reached at cs143-sum1112-staff@lists.stanford.edu. Please don't hesitate to send us emails! We're here because we genuinely love this material and want to help you learn it, so if there's anything we can do to make your life easier please let us know.

By virtue of enrolling in CS143, you should automatically be added to the student mailing list cs143-sum1112-students@lists.stanford.edu. Outside of lecture, this will be the main way that we will be making announcements.

Office Hours The three of us will be holding office hours throughout the quarter. We will be sending out an email soon where you can let us know what times would work best for you, and will try to schedule office hours to ensure that there's a time that fits nicely into your schedule.

Lectures Mondays, Wednesdays, and Fridays, 11:00AM – 12:15PM in Thornton 102.

Units This course is offered for both 3 and 4 units. If you are an undergraduate, you should be taking this course for four units, but if you're a grad student you can take it for either three or four units, depending on what makes your schedule work out better. The course is identical either way.

Prerequisites This course has CS103 and CS107 as prerequisites. From CS107, you should feel comfortable working in a Unix environment and should know how to write, debug, and test code from the command-line. Additionally, you should feel comfortable developing and testing nontrivial software systems – after all, you'll be building one over the course of the quarter! You should also have some exposure to an assembly language.

In addition to the coding background from CS107, you'll need a fair amount of theory from CS103. I expect you to be familiar with finite automata (DFAs and NFAs), regular expressions, and context-free grammars. While you don't need to be an expert on the pumping lemma or closure properties, you should feel comfortable following nontrivial constructions involving automata and grammars. We'll be substantially leveraging off this material in the first half of the course as we talk about the exciting world of scanning and parsing, so if you haven't seen these topics please come talk to us. We'd be happy to provide you some references to help you get up to speed.

The projects for this course use the C++ programming language. While you don't need to be a C++ wizard to complete the assignments, you should feel comfortable working with the language, and in particular should know how to program with classes and inheritance. If you'd like a C++ refresher, there are a bunch of useful links on the course website to help you get up to speed. I hope that the use of C++ in this course doesn't deter you from taking it – we'll be happy to help you with some of C++'s trickier points. We will also hold a C++ review session early in the quarter; location and time TBA.

Website

The course website is cs143.stanford.edu and it's loaded with resources for this course. There, you'll find all the handouts for this course, along with lecture slides and any code examples that we write in class. It also has useful links for learning more about the tools we'll be using (flex and bison), the C++ programming language, and compilers in general.

Readings

The readings for this course will mostly be in the form of handouts that are periodically distributed in-class. All the handouts will be available online, so if you miss a class or are taking the course remotely, don't worry, everything should be posted just before class starts.

In addition to the handouts, there are two books that I strongly suggest that you pick up. Neither of these textbooks are required for the course, but they're great resources. Those texts are

Compilers: Principles, Tools, and Techniques, Second Edition by Aho, Lam, Sethi, and Ullman. This book, affectionately called the "Purple Dragon Book" by its readers, is an excellent introduction to the practical and theoretical techniques necessary to build a fully-working compiler. The *Compilers* book goes into great detail about the techniques we'll be covering, then continues onward to explore more advanced concepts, many of which you can learn more about in CS243.

Parsing Techniques: A Practical Guide, Second Edition by Grune and Jacobs. Although we will only spend a few weeks talking about parsing, it's one of the most interesting and theoretically beautiful topics of the course. This book on parsing explores numerous algorithmic and theoretical techniques in parsing and has perhaps the best treatment on the subject of any text. Although narrower in scope than *Compilers*, *Parsing Techniques* does a fantastic job making parsing accessible and interesting. If you are on Stanford's network, you can view this book online through SpringerLink.

Grading

At its core, CS143 is a course about building compilers. Accordingly, this course will primarily revolve around four programming projects that taken together form a complete working compiler. Additionally, there will be two written problem sets that explore the theoretical content of the course in more detail. Taken together, you'll understand compilers from both a practical and theoretical perspective.

This course also has a midterm exam that will test your knowledge of both the theoretical and applied aspects of compiler design. I've tentatively scheduled the midterm for **Wednesday, July 25** from 11:00AM – 1:00PM, which is the regular class period extended by 45 minutes. If this is a problem, please let me know – we'd be happy to set up an alternative time. If you're taking this class over SCPD, then you'll need to take the exam over some two-hour period starting between Wednesday, July 25 at 11:00AM and Thursday, July 26 at 11:00AM. Please send your completed exam in so that it arrives no later than 1:00PM on Thursday, July 26.

Overall, your grade for this course will be determined as

Programming Projects:	**60%**
Written Assignments:	**20%**
Midterm:	**20%**

The first two programming assignments are smaller than the latter two, and so they will be weighted less.

Late Policy

The four programming projects in this course each implement one step in the compilation process, and consequently each assignment builds off of the previous one. This means that if you start falling behind on one of the assignments, it can be extremely difficult to get caught up again. That said, I completely understand that you may need some extra time to complete each of the assignments – after all, the weather really is beautiful at this time of year! In that vein, you are allowed **five** calendar late days that you can use on any of the assignments, including the written assignments, without penalty. Although you don't need to let us know that you're using them, we would appreciate a heads-up so that we know to expect your solution late. However, once you've used all five late days, your assignments will be penalized 10% per hour late, so please do try to get them in on time!

For logistical reasons, there are no late days allowed on the final programming assignment.

Honor Code

This one should be pretty simple. Don't turn in someone else's assignment as your own, or share your solution with other students. You should feel free to talk about the problem sets or programming projects in a group, but all the work you submit should be your own. That is, you should be writing up your own problem set solutions and implementing your programming projects on your own. If you do discuss ideas with other students, that's fine, please make a note of it in your submission.

In the past, this course has allowed students to work in pairs on the programming assignments. However, at least for the time being, this quarter you must complete all of the assignments individually. I may revise this policy later in the quarter, in which case I'll make an announcement to the class.