Stuff you need to know about CS144 (Fall 2025)

CS144 Staff

Prerequisites

The formal prerequisite for CS144 is CS111. CS144 is a lab-based systems course: 45% of your grade is based the programming lab in C++, which means you need to be very comfortable with C++ and using standard debugging tools (if you are reading through thousands of lines of `printf` output you are doing something wrong). CS107 is not sufficient preparation: you need more programming experience. CS144 is offered every year, so please wait until you are ready before taking it. That being said, if you did pretty well in CS110/CS111 you should do fine in CS144.

Summary

CS144 is an introductory course about computer networks. You will learn about the basic *principles* of computer networks, for example packet switching, layering, encapsulation and protocols; and you will learn how applications such as the world-wide-web, video streaming (e.g. Netflix and Hulu), video conferencing (e.g. Zoom and Skype) and BitTorrent use the network to communicate. You will spend quite a lot of time learning about the specifics of how the Internet works – which is of course by far the biggest computer network ever built. You will learn how applications communicate reliably over an unreliable Internet. And you will build portions of the Internet yourself! In fact, we believe that in CS144 you build more parts of the Internet infrastructure than in any other undergraduate networking class anywhere. It's really fun to see how the individual pieces work: You build an Internet router, and a reliable data delivery service, and then you use it to communicate with remote servers.

In addition to lectures, we will also have a few in-class guest lectures by outside speakers. All the guest lecturers are excellent speakers with many years of experience making networks work at huge scale. We will also have one or more in-class exercises, which you will complete during the regular lecture time. These are designed to give you hands-on experience with tools that are useful for your labs.

Course Website

The course website can be found at https://cs144.stanford.edu.

Credits and Workload

If you are an undergraduate, you must enroll for 4 credits. Graduate students may enroll for either 3 or 4 credits. 4 credits means that we expect you to spend, on average, approximately 12 hours per week on CS144. This time is not uniform through the quarter. But generally speaking, we expect that you'll spend:

- 2.5 hours/week in lectures,
- 2.5 hours/week in the lab session
- 7 additional hours/week working on the lab assignment or exam studying,

for a maximum of 12 hours. You should expect to spend about 70 hours total on the lab. The lab assignment is designed with the expectation that you will attend the Wednesday-evening lab sessions and work with others and in conversation

with the course staff; however, lab attendance is not mandatory for students who are up-to-date in submitting their lab checkpoints.

Weekly Units and Quizzes

The course is organized into units, with each unit lasting about a week. For example, Unit 1 is about the basic principles of networking. Each week will have three lectures (Monday, Wednesday, Friday), with a short quiz posted each Thursday and due each Friday. The goal of these quizzes is not to be extremely difficult material that requires a lot of effort, but rather simple thought experiments that lead you to think through the material a bit and help the course staff understand how well students are learning.

Website: You can reach the course website, with a pointer to where you can find all the course materials, including videos, at: `https://cs144.stanford.edu/`.

Textbook: The course has three optional texts, listed on the website (Kaashoek & Saltzer, Kurose & Ross, or Peterson & Davis). We will not be assigning readings, but these are excellent references to supplement the lectures and videos.

Class Attendance

You should try to come to all live lectures, if possible (and if space allows). You will learn more material that way; we have found that students learn more by taking part live, rather than simply watching on video. You will get more out of the class if you do.

Exams

We will have two exams: A midterm exam on September 29 (in the lab-session slot) and a final exam on December 10. We will explain how the exams work in Week 1.

Grading

Our priority in teaching this class is **your learning**. We really want to help you learn the fundamentals of computer networking, and **that's why I'm here**—not principally to evaluate you.

For those of you who are taking this course for a grade visible to external audiences, we'll do our very best to honestly evaluate and summarize how well you have achieved the learning goals of the course.

Your grade would be based on the lab assignment (45%), your quizzes and assistance to others' learning (15%), and the midterm (20%) and final (20%) exams.

- For the **lab assignment** (45%), the lab is divided into eight "checkpoints." The entire lab will receive one **correctness grade**, evaluated at the end of the class. This accounts for 27% of your final grade. In addition, checkpoints that are handed in on time will be **graded subjectively for soundness and style** by the teaching staff. The **best six** soundness/style grades will contribute 18% to the final grade (3% for each checkpoint).
- For the in-class quizzes, the **best six** will contribute 12% of your final grade. An additional 3% of your final grade will be determined by helping others learn in the class (e.g., answering questions well on EdStem, being credited in others' lab writeups, asking good questions in lecture and EdStem). Please do not neglect this portion of the class.

If you feel you were graded incorrectly on a homework, lab, or exam, please let us know as soon as possible. At the end of the quarter, we take the distribution of numerical grades and decide what ranges correspond to what letter grades. We don't decide on grade ranges *a priori* because sometimes exam questions are harder or easier than we thought they would be, and so we want to be able to adjust accordingly. However, CS144 is not graded on a curve: we decide on grade ranges, not class population percentages. We do not publish the numerical grade ranges corresponding to letter grades.

Late Policy

Programming lab checkpoint deadlines are set to give us enough time to discuss them at our course meeting and grade them and give feedback in a timely manner. We cannot guarantee that late submissions will be graded for soundness/style.

If you struggle with Checkpoints 0 or 1 because you find it difficult to write the code, you might want to consider dropping the class. The first two checkpoints will give you early feedback on whether your programming skills are sufficient for the course.

If a real-life event (wedding, funeral, COVID, hospitalization, etc.) disrupts your ability to turn an assignment in on time, please let us know as early as possible. Clearly, some such events, such as a trip to the emergency room, are less expected than others, and we understand. Emailing the staff 48 hours before an assignment is due asking for an extension because you have a wedding to go to might be met with a frown; email us two weeks before the assignment is due and we'll do our best to accommodate. Our goal is to make sure you don't fall behind on the next assignment.

Incomplete Policy

Our policy is not to give incompletes for CS144 except in extraordinary circumstances. If you are falling behind or something life-changing comes up, please contact us immediately and we'll try to work something out. Contacting us early is better than late. Generally, taking too heavy a course load is not a sufficient justification: courses last a quarter for a reason and you are expected to be responsible for your own schedule. We don't allow incompletes because grading programming assignments outside the normal quarter is exceedingly difficult (and inconsistent).

Office Hours and Email

If you have a question about the class material or a programming assignment, please ask on EdStem, in the lab session, or in office hours. Here are some guidelines on how you can ask questions to maximize the amount and quality of help we can provide.

Please use EdStem for questions about programming assignments and general course questions. Using EdStem means that everyone can benefit from the answer; it may be that other students had the same question. Please do ask questions about the requirements of the assignment, the provided code, or the expected behavior of your system. Please don't ask questions that relate to how to implement a solution. For example, please don't ask questions that include or ask for source code. If you have any uncertainty about whether a question is OK, please email the course staff. You can find answers to almost any general C++ question on the web.

If you have questions about your particular solution to an assignment, you should come to office hours, or ask on an evening lab session. For personal questions (e.g., arranging an appointment, questions about grading), please send an email cc'ing both instructors. Email is better than office hours for questions on grading because it may be the staff member at office hours wasn't the one who graded your assignment.

Generally speaking, it's almost impossible to answer programming assignment questions over email. The round-trip-time is too long, and it's not interactive. Email discussions often boil down to needing a TA to find a bug for you, which isn't very educational. Therefore, please attend lab sessions or office hours to discuss programming questions.

Honor Code

We report all Honor Code Violations. We use tools to catch plagiarizers and the punishments are severe. Honor Code violations are taken very seriously and we want to encourage you to never venture over the line. Our goal for this year is for zero violations of the Honor Code. In 2016 and 2017, we succeeded. But in 2018 and 2019 we had over five violations. We would love to have a quarter free of any Honor Code Violations.

Most Honor Code Violations in CS144 happen when students reuse and modify code written by other students, including solutions posted on-line. In this course, we take the Honor Code very seriously and we expect all students

to do the same. The good news is that the vast majority of students do follow the Honor Code. The bad news is that historical evidence indicates that some students will submit work that is not their own, not only shortchanging their own learning, but undermining the atmosphere of trust and individual achievement that characterizes Stanford's academic community. To protect academic integrity and the interests of all students, the course staff will investigate all possible Honor Code violations and refer them to the Office of Community Standards as necessary.

If you have any questions or doubts about the Honor Code, please talk to the instructors. Honor Code violations are no laughing matter at Stanford and it is much better to ask what might seem like a silly question now than risk your academic career. The Honor Code has a long tradition at Stanford dating back to Spring 1921 when the University first adopted the honor system.

The underlying premise of the policy is that all academic work represents independent, original work of the author and the Honor Code aims to foster an academic environment that encourages adherence to these principles. As we are all bound to respect and uphold the Honor Code, it is important to define acceptable and unacceptable behaviors with regard to this course so as to eliminate any ambiguity.

Permitted Collaboration: The following items are encouraged and allowed at all times for all students in this class:

- Discussion of material covered during lecture, problem sessions, or in handouts
- Discussion of the requirements of an assignment
- Discussion of the use of tools or development environments
- Discussion of general approaches to solving problems
- Discussion of general techniques of coding or debugging
- Discussion between a student and a TA or instructor for the course

Collaboration Requiring Citation: Two students engaging in more detailed discussions must be careful to document their collaboration. Students are required to include the names of those who provide specific assistance to properly credit their contribution, in the same manner as one would cite a reference in a research paper. The expectation is that even with a citation, the author must be able to explain the solution. Some examples of collaboration that require citation include:

- Discussing the "key" to a problem set or programming assignment. Problem set questions are often designed such that the critical concept takes careful thought and gaining that insight from someone else must therefore be documented.
- Discussing the design of a programming project. Design is a crucial aspect of the programming process and discussion can be valuable. Any design input received from others must be cited.
- Receiving assistance from another student in debugging code. While the TAs are the preferred source for advice, any detailed assistance from someone else must be credited.
- Sharing advice for testing. For example, if someone provides important information on lessons learned ("my program didn't handle the case where the value was 0") that source must be credited.
- Research from alternative sources. Researching related topics, such as through the Internet, must be documented if the solution submitted is derived from the research information.

Unpermitted Collaboration: All submissions must represent original, independent work. Some examples of activities that do not represent original work include:

- Copying solutions from others or knowingly allowing others to copy your solution. In particular, do not ask anyone to provide a copy of his or her solution or, conversely, give a solution to another student who requests it. Similarly, do not discuss algorithmic strategies to such an extent that you and your collaborator submit exactly the same solution. Use of solutions posted to websites, such as at other universities, is prohibited. Be aware that we photocopy some of the exams prior to handing them back. Also be aware that **placing your source code for the course in a publicly accessible repository where others can copy it is unpermitted collaboration.**

- Using work from past quarters. The use of another student's solution or the posted class solutions from a previous quarter constitutes a violation. We use a sophisticated software tool that cross-checks every assignment against every other assignment submitted this year, and previous years. It catches common code, even if comments and variable names are changed. In fact, in order to "fool" it, you have to change so much code that it would be quicker to do the assignment yourself (we tried it!). Developing good problem set questions and programming assignments often takes years and new assignments invariably have problems and that require polishing. To provide the most effective exercises, questions and assignments are commonly reused. Students retaking the course are expected to notify the course staff to avoid coming under suspicion. If you looked at solutions before taking the course, delete any such code you might have and email the course staff mailing list immediately to let us know.
- Studying another student's solution. Do not read another solution submission whether in electronic or printed form, even to "check answers."
- Debugging code for someone else. When debugging code it is easy to inadvertently copy code or algorithmic solutions. It is acceptable to describe a problem and ask for advice on a way to track down the bug. "What would you do to try to find this bug?" is an acceptable question; "Can you help me find my bug??" is not.
- Collaborating on or discussing the online graded quizzes before you have completed them. These are intended to be relatively easy, simple questions that test your basic knowledge.
- For purposes of these guidelines, chatbots (e.g. ChatGPT, Gemini) should be considered equivalent to "a student who has already taken this course."

This section on the Honor Code was based on policies written by Tom Fountain, Eric Roberts, Julie Zelenski, and the Computer Science Department at Brown University.