Description

Introduction to the intellectual enterprises of computer science and to the art of programming. Students learn how to think algorithmically and solve problems efficiently. Topics include abstraction, algorithms, data structures, encapsulation, resource management, security, software engineering, and web development. Languages include C, Python, SQL, and JavaScript, plus CSS and HTML. Problem sets inspired by real-world domains of biology, cryptography, finance, forensics, and gaming. See CS50's website, cs50.yale.edu, for additional information. No previous programming experience required. Open to students of all levels and majors.

Expectations

You are expected to

- watch all lectures,
- attend eight sections,
- solve eight problem sets,
- submit eight checks for understanding, and
- design and implement a final project.

Learning Objectives

Among this course's objectives are that you learn how to:

- think more methodically;
- program procedurally;
- represent and process information;
- communicate succinctly and precisely;
- solve problems efficiently;
- recognize patterns among problems;
- decompose problems into parts and compose solutions thereto;
- operate at multiple levels of abstraction;
- separate design from implementation details;
- infer from first principles how systems work;
- assess the correctness, design, and style of code;
- teach yourself new languages;
- identify threats to privacy and security;
- read documentation, drawing conclusions from specifications;
- test solutions to problems, find faults, and identify corner cases;
- describe symptoms of problems precisely and ask questions clearly; and
- identify and quantify trade-offs among resources, particularly time and space.

Ultimately, the course aspires to provide you with a foundation for further studies in computer science and to empower you to apply computer science to problems in other domains.

Outline

Outlined below is the course's subject matter, organized by week, each subtitled per to the context in which its topics are introduced.

Week 1 Intro to C & Arrays

- C. Source Code. Machine Code. Compiler. Correctness, Design, Style. Visual Studio Code. Syntax Highlighting. Escape Sequences. Header Files. Libraries. Manual Pages. Types. Conditionals. Variables. Loops. Linux. Graphical User Interface (GUI). Command-Line Interface (CLI). Constants. Comments. Pseudocode. Operators. Integer Overflow. Floating-Point Imprecision.
- Preprocessing. Compiling. Assembling. Linking. Debugging. Arrays. Strings. Command-Line Arguments. Cryptography.

Week 2 Algorithms & Memory

- Searching: Linear Search, Binary Search. Sorting: Bubble Sort, Selection Sort, Merge Sort. Asymptotic Notation, Recursion.
- Pointers. Segmentation Faults. Dynamic Memory Allocation. Stack. Heap. Buffer Overflow. File I/O. Images.

Week 3 Python & SQL

- Python: Functions, Arguments, Return Values; Variables; Boolean Expressions, Conditionals; Loops. Modules, Packages.
- SQL: Tables; Types; Statements; Constraints; Indexes; Keywords, Functions; Transactions. Race Conditions. SQL Injection Attacks.

Week 4 HTML, CSS, JavaScript & Flask

- Internet: Routers; TCP/IP; DNS. HTTP: URLs, GET, POST. HTML: Tags; Attributes. Servers. Regular Expressions. CSS: Properties; Selectors. Frameworks. JavaScript: Variables; Conditionals; Loops. Events.
- Flask. Route. Decorators. Requests, Responses. Sessions. Cookies.

Grades

You are encouraged to take CS50 Credit/D/Fail if you will feel less nervous without the pressure of a letter grade. Note it will only satisfy the QR requirement if you take it for a grade. Please also remember that your course grade will reflect how much you progress over the semester. We expect you to work hard and learn a lot, and your course grades will reflect that. Course grades tend to be quite high in CS50.

Whether taking the course Credit/D/Fail or for a letter grade, you must ordinarily submit all problem sets, checks for understanding, one test, and submit a final project unless granted an exception in writing by the course's heads. Multiple missing problem sets, a missing check for understanding or final project, and violations of the academic honesty policy may each result in a reduced or failing grade.

Graduate and professional students are expected to produce a final project that is 50% greater in scope than what is expected of undergraduate students. Your course grade will be adapted to the prevailing standards of your school, but we encourage you to take the class Pass/Fail if your degree program permits it.

Final grades are determined using the following weights:

Problem Sets	50%
Checks for Understanding	10%
Final Project	20%
Attendance*	20%

* At sections.

Books

No books are required or recommended for this course.

Sections

Lectures are supplemented by in-class discussion sections held twice weekly. Attendance at sections is expected.

Problem Sets

Problem sets are programming assignments that allow you to implement each lecture's concepts in code.

Late work is not ordinarily accepted, except with a Dean's excuse. See Lateness Policy for more information. (Graduate students should e-mail heads@cs50.yale.edu as early as possible to request an extension.)

Checks for Understanding

Checks for understanding ("checks") are short assignments due after each lecture that ask you to practice applying each lecture's concepts. Among the goals of the checks for understanding are that you synthesize knowledge from class alone. You may use any materials on CS50's course website (i.e., cs50.harvard.edu), but nothing else (i.e., you may not use tools like cs50.ai or cs50.dev, nor search for information on the internet at large). The only humans to whom you may turn for help or from whom you may receive help are the course's heads. Your 7 highest scores will be counted towards your final grade.

Final Project

The climax of this course is its final project. The final project is your opportunity to take your newfound savvy with programming out for a spin and develop your very own piece of software. So long as your project draws upon this course's lessons, the nature of your project is entirely up to you, albeit subject to the staff's approval. You may implement your project in any language(s) as long as the staff approves. You are welcome to utilize any infrastructure, provided the staff ultimately has access to any hardware and software that your project requires. All that we ask is that you build something of interest to you, that you solve an actual problem, that you impact campus, or that you change the world. Strive to create something that outlives this course.

Inasmuch as software development is rarely a one-person effort, you are allowed an opportunity to collaborate with one or two classmates for this final project. Needless to say, it is expected that every student in any such group contribute equally to the design and implementation of that group's project. Moreover, it is expected that the scope of a two- or three-person group's project be, respectively, twice or thrice that of a typical one-person project. A one-person project, mind you, should entail more time and effort than is required by each of the course's problem sets. Although no more than three

students may design and implement a given project, you are welcome to solicit advice from others, so long as you respect the course's policy on academic honesty.

Extensions on the final project always require a Dean's excuse (graduate students must provide documentation of a genuine emergency). Late submissions will receive no credit.

Milestone	Deadline
Proposal	TBD
Status Report	TBD
Implementation	TBD

CS50 Fair

The CS50 Fair is an epic display of final projects. Not only is the CS50 Fair a venue at which to see classmates' projects and demo your own, it is an opportunity to mingle with students, faculty, and staff from across campus.

Attendance is expected of all students.

Also in attendance are popcorn, candy, and a raffle with (fabulous) prizes. Family and friends are welcome to join.

Lateness

We do not accept late checks for understanding or labs except in cases of serious, multi-week illnesses and emergencies that receive a dean's excuse. However, we will drop the two lowest/missed checks for understanding to account for shorter illnesses and other problems.

Late pset submissions are not accepted without a dean's excuse. However, we will allow a *single* 72 hours pset extension for all undergraduate students, known as the brink clause. The brink clause must be invoked prior to the pset deadline and cannot be revoked. To use this extension, fill out this form, and please use this as an opportunity to let us know if you have been struggling in the course. We will follow up with an email and a meeting to help you get back on track.

Graduate students must receive an extension in advance from instructors. In addition to documented illness and emergencies, we recognize that graduate programs often have a week sometime during the semester when students must dedicate all their time to a single project or activity. We therefore allow graduate students a *single,* one-week pset extension provided it is requested by at least 72 hours before the deadline by e-mail to heads. This does not apply to checks for understanding, which are covered by the policy above.

Academic Honesty

This course's philosophy on academic honesty is best stated as "be reasonable." The course recognizes that interactions with classmates and others can facilitate mastery of the course's material. However, there remains a line between enlisting the help of another and submitting the work of another. This policy characterizes both sides of that line.

The essence of all work that you submit to this course must be your own and you must explicitly cite anyone you collaborate with and any resources you use that are not part of the course material or directly linked from the pset instructions.

> You may ask classmates and others for help on psets, and you may use outside resources that do not reduce to another doing your work for you. You must document via comments at the top of your related code file the full name of the person you discussed with ("John Q. Adams" is good, "my roommate," "my tutor," or "Natalie" is not), or the complete URL or reference for the outside resource; what you discussed, and how that helped you.
> You must also cite any help or discussions you have with other students in the class in your solution comments.
> You may not post your code online, and you may not look at or use online solutions to the psets.
> Collaboration on the course's checks for understanding is not permitted at all.
> Collaboration on labs is permitted in pairs *during section,* provided it is documented in your program comments. Further work on labs outside section must be individual. Collaboration on the course's final project is permitted only to the extent prescribed by its specification.

Regret clause. If you commit some act that is not reasonable but bring it to the attention of the course's heads within 72 hours, the course may impose local sanctions that may include an unsatisfactory or failing grade for work submitted, but the course will not refer the matter for further disciplinary action unless another infraction occurs. You may invoke this clause only once for a true infraction, but if you self-report something that the

course heads feel is not a significant violation, it will not count against you or count as your one invocation.

In cases of suspected violations involving students at both Harvard and Yale, students will be referred to the appropriate committee at their university. Those committees may exchange information for the purpose of resolving the cases in accordance with their own procedures. They may also reach different conclusions and impose different sanctions from the same set of facts and evidence.

Below are rules of thumb that (inexhaustively) characterize acts that the course considers reasonable and not reasonable. If in doubt as to whether some act is reasonable, do not commit it until you solicit and receive approval in writing from the course's heads. Acts considered not reasonable by the course are handled harshly. If the course refers some matter for disciplinary action and the outcome is punitive, the course reserves the right to impose local sanctions on top of that outcome that may include an unsatisfactory or failing grade for work submitted or for the course itself.

Reasonable

- Communicating with classmates about problem sets' problems in English (or some other spoken language), and properly citing those discussions.
- Discussing the course's material with others in order to understand it better. You do not need to cite this if it isn't related to the pset, but we encourage citing and note-taking on your discussion anyway.
- Helping a classmate identify a bug in their code at office hours, elsewhere, or even online, as by viewing, compiling, or running their code after you have submitted that portion of the pset yourself. Add a citation to your own code of the help you provided and resubmit.
- Incorporating a few lines of code that you find online or elsewhere into your own code, provided that those lines are not themselves solutions to assigned problems and that you cite the lines' origins in a comment in your code and in your program comments.
- Reviewing past semesters' tests and quizzes and solutions thereto that are made available by the course. (You do not need to cite this; it is considered part of the standard course materials.)
- Sending or showing code that you've written to someone, possibly a classmate, so that he or she might help you identify and fix a bug, provided you properly cite the help. If it is a classmate, make sure they cite giving the help as well.
- Submitting the same or similar work to this course that you have submitted previously to this course, CS50 AP, or CS50x, so long as you disclose as much in your submission, as via comments in your code. Please include a comment at

the top of the file indicating it is a resubmission to eliminate any potential confusion.
Turning to the course's heads for help or receiving help from the course's heads with the checks for understanding.
Turning to the web or elsewhere for instruction beyond the course's own, for references, and for solutions to technical difficulties, but not for outright solutions to problem set's problems or your own final project. Cite any such resources in your program comments, especially if they are related to a pset.
Using CS50's own AI-based software, including the CS50 Duck (ddb) in cs50.ai and cs50.dev as well as in Ed.
Whiteboarding solutions to problem sets with others using diagrams or pseudocode but not actual code. Cite who you discussed with and what you discussed in your program comments, even if you are doing this in office hours under TA supervision. It helps avoid misunderstandings down the road.
Working with (and even paying) a tutor to help you with the course, provided the tutor does not do your work for you and you cite the pset help you receive in your program comments.

Not Reasonable

Accessing a solution to some problem prior to its deadline or (re-)submitting your own.
Accessing or attempting to access, without permission, an account not your own.
Asking a classmate to see their solution to a problem set's problem before its deadline or (re-)submitting your own.
Decompiling, deobfuscating, or disassembling the staff's solutions to problem sets.
Discovering but failing to disclose to the course's heads bugs in the course's software that affect scores.
Decompiling, deobfuscating, or disassembling the staff's solutions to problem sets.
Failing to cite (as with comments) the origins of code or techniques that you discover outside of the course's own lessons and integrate into your own work, even while respecting this policy's other constraints.
Giving or showing to a classmate a solution to a problem set's problem when it is he or she, and not you, who is struggling to solve it.
Looking at another individual's work on the quizzes.
Manipulating or attempting to manipulate scores artificially, as by exploiting bugs or formulas in the course's software.
Paying or offering to pay an individual for work that you may submit as (part of) your own.

Providing or making available solutions to problem sets to individuals who might take this course in the future.
Searching for or soliciting outright solutions to problem sets online or elsewhere.
Splitting a problem set's workload with another individual and combining your work.
Submitting (after possibly modifying) the work of another individual beyond the few lines allowed herein.
Submitting the same or similar work to this course that you have submitted or will submit to another (non-CS50) course.
Submitting work to this course that you intend to use outside of the course (e.g., for a job) without prior approval from the course's heads.
Turning to humans (besides the course's heads) for help or receiving help from humans (besides the course's heads) on the checks for understanding or test.
Using AI-based software other than CS50's own (e.g., ChatGPT, GitHub Copilot, Bing Chat, et al.) that suggests or completes answers to questions or lines of code.
Viewing another's solution to a problem set's problem and basing your own solution on it.
Viewing the solution to a lab before trying to solve it yourself.