Syllabus

This is CS50. Harvard University. Fall 2014.

Table of Contents

version 0

Instructor

David J. Malan
malan@harvard.edu[1]
http://cs.harvard.edu/malan

Description

Introduction to the intellectual enterprises of computer science and the art of programming. This course teaches students 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, PHP, and JavaScript plus SQL, CSS, and HTML. Problem sets inspired by real-world domains of biology, cryptography, finance, forensics, and gaming. Designed for concentrators and non-concentrators alike, with or without prior programming experience.

Expectations

You are expected to submit nine problem sets and a final project.

Website

This course lives at http://cs50.edx.org/.

Visit the course's website to watch videos, to get help, to download handouts and software, and to follow links to other resources.

Certificates

Students who earn a satisfactory grade (60% or higher) on every one of nine problem sets and on a final project will be eligible to receive a certificate from HarvardX as a downloadable, printable PDF.

[1] mailto:malan@harvard.edu

Books

No books are required for this course. However, you may want to supplement your preparation for or review of some lectures with self-assigned readings relevant to those lectures' content from either of the books below. The first is intended for those inexperienced in (or less comfortable with the idea of) programming. The second is intended for those experienced in (or more comfortable with the idea of) programming.

For Those Less Comfortable

C Programming Absolute Beginner's Guide, Third Edition
Greg Perry, Dean Miller
Pearson Education, 2014
ISBN 0-789-75198-4

For Those More Comfortable

Programming in C, Fourth Edition
Stephen G. Kochan
Pearson Education, 2015
ISBN 0-321-77641-0

The book below is recommended for those interested in understanding how their own computers work for personal edification.

How Computers Work, Ninth Edition
Ron White
Que Publishing, 2008
ISBN 0-7897-3613-6

This last book below is recommended for aspiring hackers, those interested in programming techniques and low-level optimization of code for applications beyond the scope of this course.

Hacker's Delight, Second Edition
Henry S. Warren Jr.
Pearson Education, 2013
ISBN 0-321-84268-5

Lectures

A schedule of lectures, subject to change, appears below.

Week 0

Binary. ASCII. Algorithms. Pseudocode. Source code. Compiler. Object code. Scratch. Statements. Boolean expressions. Conditions. Loops. Variables. Functions. Arrays. Threads. Events.

Week 1

Linux. C. Compiling. Libraries. Types. Standard output.

Week 2

Casting. Imprecision. Switches. Scope. Strings. Arrays. Cryptography.

Week 3

Command-line arguments. Searching. Sorting. Bubble sort. Selection sort. Insertion sort. O. Ω .Θ. Recursion. Merge Sort.

Week 4

Stack. Debugging. File I/O. Hexadecimal. Strings. Pointers. Dynamic memory allocation.

Week 5

Heap. Buffer overflow. Linked lists.

Week 6

Hash tables. Tries. Trees. Stacks. Queues.

Week 7

TCP/IP. HTTP. HTML. CSS.

Week 8

PHP. MVC. SQL.

Week 9

JavaScript. Ajax.

Week 10

Security. Guest Lecture.

Week 11

holiday

Week 12

Exciting conclusion.

Sections

Lectures are supplemented by weekly, 90-minute sections led by the teaching fellows. Sections provide you with opportunities to explore the course’s material in a more intimate environment.

Postmortems

Available after problem sets' deadlines are "postmortems," videos via which the course’s staff explore actual solutions to problem sets. You are expected to watch postmortems for insights into how else you could have (or should have!) implemented your own solutions.

Problem Sets

Nine problem sets are assigned during the semester. Each is due by 31 December 2015.

In order to accommodate students with different backgrounds, some problem sets are released in two editions: a standard edition intended for most students and a "Hacker

Edition" intended for some students. Both editions essentially cover the same material. But the Hacker Edition typically presents that material from a more technical angle and poses more sophisticated questions. **To receive an honor code certificate from HarvardX[2], you must submit the standard editions of problem sets. You are welcome to do the Hacker Editions for your own edification, but it is not possible to submit Hacker Editions for credit (or extra credit).**

A schedule of problem sets appears below.

- Problem Set 0: Scratch
- Problem Set 1: C
- Problem Set 2: Crypto
- Problem Set 3: Breakout
- Problem Set 4: Forensics
- Problem Set 5: Mispellings
- Problem Set 6: Web Server
- Problem Set 7: C$50 Finance
- Problem Set 8: Mashup

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. You may implement your project in any language(s), and you are welcome to utilize infrastructure other than the CS50 Appliance. All that we ask is that you build something of interest to you, that you solve an actual problem, 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

[2] https://www.edx.org/school/harvardx

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.

See http://cdn.cs50.net/2015/x/project/project.html for details.

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. Collaboration on problem sets is not permitted except to the extent that you may ask classmates and others for help so long as that help does not reduce to another doing your work for you. Generally speaking, when asking for help, you may show your code to others, but you may not view theirs, so long as you and they respect this policy’s other constraints. Collaboration on the course’s final project is permitted to the extent prescribed by its specification.

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.

Reasonable

- Communicating with classmates about problem sets' problems in English (or some other spoken language).
- Discussing the course’s material with others in order to understand it better.
- Helping a classmate identify a bug in his or her code at Office Hours, elsewhere, or even online, as by viewing, compiling, or running his or her code, even on your own computer.

- Incorporating snippets of code that you find online or elsewhere into your own code, provided that those snippets are not themselves solutions to assigned problems and that you cite the snippets' origins.
- Reviewing past semesters' quizzes and solutions thereto.
- 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.
- Sharing snippets of your own code online so that others might help you identify and fix a bug.
- 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.
- Whiteboarding solutions to problem sets with others using diagrams or pseudocode but not actual code.
- Working with (and even paying) a tutor to help you with the course, provided the tutor does not do your work for you.

Not Reasonable

- Accessing a solution in CS50 Vault to some problem prior to (re-)submitting your own.
- Asking a classmate to see his or her solution to a problem set’s problem before (re-)submitting your own.
- 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 during a quiz.
- 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, soliciting, or viewing a quiz’s questions or answers prior to taking the quiz.
- 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 allowed snippets.
- Submitting the same or similar work to this course that you have submitted or will submit to another.
- 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.
- Using resources during a quiz beyond those explicitly allowed in the quiz’s instructions.
- Viewing another’s solution to a problem set’s problem and basing your own solution on it.