Fundamentals of Computer Science and Programming

(CS50's Introduction to Computer Science from Harvard University)

Syllabus (Spring 2023)

Please read this sheet carefully and save it for future reference.

Instructors Name	Email
Ilja Afanasjevs	ilja.afanasjevs@rbs.lv
Maris Purvins	maris.purvins@rbs.lv

Course Identification

Course Name: Fundamentals of Computer Science and Programming
Course Type: LEC (Online), LAB/Workshops
Meeting location: Riga Business School, Alumni Auditorium, Skolas st. 11, Riga.
Meeting times: Thursdays, **18:00 - 20:00**, from 9th of March.
Office Hours: Online via Zoom.
Study Materials Adopted: CS50's Introduction to Computer Science from Harvard University
Language: English
Lectures: Harvard Provided Video Lectures, Harvard's Computer Science Professor David J. Malan
RTU Course code: DIP750
Credit Hours: 4.0 Credit Points, 6.0 ECTS credits

Registration

Students from **any Latvian Higher Education Institution can apply** for this course. To apply for this course, please fill out this form: ej.uz/cs50latvia or scan QR code:

Cabinet of Ministers Republic of Latvia

Cabinet of Ministers Regulation No. 315. Regulations Regarding the Implementation of the First, Second, and Third Project Application Selection Round for the Activity 1.1.1.5, "Support for International Cooperation Projects in Research and Innovation".

Course Description

This is CS50 Latvia, adopted from Harvard University's introduction to the intellectual enterprises of computer science and the art of programming, for concentrators and non-concentrators alike, with or without prior programming experience. (Two-thirds of CS50 students have never taken CS before.) This course teaches you how to solve problems, both with and without code, with an emphasis on correctness, design, and style. Topics include computational thinking, abstraction, algorithms, data structures, and computer science more generally. Problem sets inspired by the arts, humanities, social sciences, and sciences. More than teach you how to program in one language, this course teaches you how to program fundamentally and how to teach yourself new languages ultimately. The course starts with a traditional but omnipresent language called C that underlies today's newer languages, via which you'll learn not only about functions, variables, conditionals, loops, and more, but also about how computers themselves work underneath the hood, memory and all. The course then transitions to Python, a higher-level language that you'll understand all the more because of C. Toward term's end, the course introduces SQL, via which you can store data in databases, along with HTML, CSS, and JavaScript, via which you can create web and mobile apps alike. Course culminates in a final project.

Course Schedule In-Class activities

Activity	Date and Time	Where
Intro Class	Thu, Mar 9, 18:00	In RBS Alumni Auditorium
Scratch Lab	Thu, Mar 16, 18:00	In RBS Alumni Auditorium
Lab 1	Thu, Mar 23, 18:00	In RBS Alumni Auditorium
Lab 2	Thu, Mar 30, 18:00	In RBS Alumni Auditorium
Lab 3	Thu, Apr 6, 18:00	In RBS Alumni Auditorium
Lab 4	Thu, Apr 13, 18:00	In RBS Alumni Auditorium
Lab 5	Thu, Apr 20, 18:00	In RBS Alumni Auditorium
Lab 6	Thu, Apr 27, 18:00	In RBS Alumni Auditorium
Lab 7	Thu, May 11, 18:00	In RBS Alumni Auditorium
Lab 8	Thu, May 18, 18:00	In RBS Alumni Auditorium
Lab 9	Thu, May 25, 18:00	In RBS Alumni Auditorium
Final Project Day	Thu, Jun 1, 18:00	In RBS Alumni Auditorium
Final Test Day	Thu, Jun 8, 18:00	In RBS Alumni Auditorium

Software/hardware and support

- DB Browser for SQL Lite (https://sqlitebrowser.org/dl/)
- Scratch (https://scratch.mit.edu/)
- Visual Studio Code (https://code.cs50.io/)
- LMS – RTU ORTUS E-Studies System (https://ortus.rtu.lv/)
- Laptop or any other portative device.

Textbook

No books are required or recommended for this course. However, you might find the below books of interest. Realize that free, if not superior, resources can be found on the course's website.

Hacker's Delight, Second Edition Henry S. Warren Jr. Pearson Education, 2013 ISBN 0-321-84268-5	How Computers Work, Tenth Edition Ron White Que Publishing, 2014 ISBN 0-7897-4984-X	Programming in C, Fourth Edition Stephen G. Kochan Pearson Education, 2015 ISBN 0-321-77641-0

Expectations

You are expected to

- attend ten LAB/Workshop sections,
- complete nine labs,
- solve ten problem sets,
- take nine quizzes in class before LAB,
- take one test in the end of the course
- 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 (date)	Topic	Content
Week 0	Scratch	Computer Science. Computational Thinking. Problem Solving: Inputs, Outputs. Representation: Unary, Binary, Decimal, ASCII, Unicode, RGB. Abstraction. Algorithms. Running Times. Pseudocode. Scratch: Functions, Arguments, Return Values; Variables; Boolean Expressions, Conditionals; Loops; Events; Threads.
Week 1	C	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.
Week 2	Arrays	Preprocessing. Compiling. Assembling. Linking. Debugging. Arrays. Strings. Command-Line Arguments. Cryptography.
Week 3	Algorithms	Searching: Linear Search, Binary Search. Sorting: Bubble Sort, Selection Sort, Merge Sort. Asymptotic Notation: Ω,O,ΘRecursion.
Week 4	Memory	Pointers. Segmentation Faults. Dynamic Memory Allocation. Stack. Heap. Buffer Overflow. File I/O. Images.
Week 5	Data Structures	Abstract Data Types. Queues, Stacks. Linked Lists. Trees, Binary Search Trees. Hash Tables. Tries.
Week 6	Python	Python: Functions, Arguments, Return Values; Variables; Boolean Expressions, Conditionals; Loops. Modules, Packages.
Week 7	SQL	SQL: Tables; Types; Statements; Constraints; Indexes; Keywords, Functions; Transactions. Race Conditionals. SQL Injection Attacks.
Week 8	HTML, CSS, JavaScript	Internet: Routers; TCP/IP; DNS. HTTP: URLs, GET, POST. HTML: Tags; Attributes. Servers. CSS: Properties; Selectors. Frameworks. JavaScript: Variables; Conditionals; Loops. Events.
Week 9	Flask	Flask. Route. Decorators. Requests, Responses. Sessions. Cookies.

Grades

Final grades are determined using the following weights:

Problem Sets	40%
Quizzes	10%
Labs	15%
Test	15%
Final Project	10%
Attendance	10%

Problem sets and the final project are evaluated along axes of correctness, design, and style, with scores ordinarily computed as 2 × correctness + 2 × design + 1 × style. Scores are normalized across teaching fellows and comfort levels at term's end, so mid-semester comparisons among students of scores are not reliable indicators of standing.

Know that CS50 draws quite the spectrum of students, including "those less comfortable," "those more comfortable," and those somewhere in between. However, what ultimately matters in this course is not so much where you end up relative to your classmates but where you end up relative to yourself when you begin.

Labs

Sections are supplemented by weekly, 75-minute labs led by the course's teaching fellows. Labs are (even smaller) opportunities to work on practice problems for the week's problem set. Here are all the lab times and locations.

Labs are planned as an in-class activity. This activity aimed for a practical introduction

Office Hours

Office hours are end-of-week online (via Zoom) opportunities for help with the problem sets alongside all of the course's teaching fellows and assistants.

Problem Sets

Activity	Programing Language	Submission Deadline
Problem Set 0	Scratch	Wed, Mar 22, 23:59
Problem Set 1	C	Wed, Mar 29, 23:59
Problem Set 2	C	Wed, Apr 5, 23:59
Problem Set 3	C	Wed, Apr 12, 23:59
Problem Set 4	C	Wed, Apr 19, 23:59
Problem Set 5	C	Wed, Apr 26, 23:59
Problem Set 6	Python	Wed, May 10, 23:59
Problem Set 7	SQL	Wed, May 17, 23:59

Problem Set 8	HTML, CSS, JavaScript	Wed, May 24, 23:59
Problem Set 9	Python, SQL, HTML, CSS, JavaScript	Wed, May 31, 23:59

Quizzes

Quizzes are short checks for understanding in first 30 minutes before the LAB. The intent of each quiz is to help you apply each week's concepts to new problems. Each quiz is open-book: you may use any and all non-human resources during a quiz, but the only humans to whom you may turn for help or from whom you may receive help are the course's heads. Quizzes are released at the start of the LAB/Workshop and are available for the first 30 minutes of the LAB.

Test

The test is an opportunity to synthesize concepts across weeks and solve new problems based on lessons learned. The test is open-book: you may use any and all non-human resources during the test, but the only humans to whom you may turn for help or from whom you may receive help are the course's heads.

Final Project

Activity	Submission Deadline
Proposal	Thu, Apr 6, 23:59
Status Report	Wed, May 3, 23:59
Fnal project submission	Wed, May 31, 23:59

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 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.

Academic Honesty

The 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 quizzes and test is not permitted at all. Collaboration on the course's final project is permitted 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 except in cases of repeated acts.

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. The course ordinarily recommends exclusion (i.e., required withdrawal) from 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.
- 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.

- Reviewing past semesters' tests and 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, provided you properly cite the help.
- 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.
- Turning to the course's heads for help or receiving help from the course's heads during the quizzes or test.
- 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 to some problem prior to its deadline.
- 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.
- 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 during the quizzes or test.
- 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.
- 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) during the quizzes or test.
- Using AI-based software that suggests or completes 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.