ISM

COMPUTER PROGRAMMING

Course code	*IT103*
Course title	*Computer Programming*
Type of course	*Elective*
Year of study	*2nd*
Semester	*Fall*
ECTS	*6 credits: 50 contact hours, 112 individual work hours*
Coordinating lecturers	*Oleg Mirzianov*
Studies form	*Full-time*
Prerequisites	-
Language of instruction	*English*

Annotation

This course is based on the Harvard CS50x course curriculum - introduction to the intellectual enterprises of computer science and the art of programming. Through the course, students learn how to solve real-life problems in the most efficient way using algorithms. The course introduces problem-solving, which is inspired by the arts, humanities, social sciences, and economics especially. No prior knowledge in programming is required. Students are expected to create a personal final project. The course will examine in depth the programming fundamentals.

Aim of the course

The aim of the course is to learn essential programming skills and get hands-on experience by solving real-life problems. The course aims the need for leaders to understand the principles of IT products' development for effective decision making.

Course Learning Outcomes (CLO)	**Study Methods**	**Assessment Methods**
CLO1. To be able to process information and operate at multiple levels of abstraction;	Lecture, consultation, problem solving, homework, discussion. self-study	Coursework, midterm, final exam, final project.
CLO2. To be able to decompose IT problems into parts and solve them efficiently;	Lecture, consultation, problem solving, homework, discussion.	Coursework, midterm, final exam, final project.
CLO3. To be able to demonstrate proficiency in a software development environment;	Lecture, consultation, problem solving, homework, discussion.	Coursework, final exam, final project.
CLO4. To be able to assess the correctness, design, and style of code;	Lecture, consultation, problem solving, homework, discussion.	Coursework, midterm, final exam, final project.
CLO5. To be able to evaluate the project's complexity and estimate required resources.	Lecture, consultation, problem solving, homework, discussion.	Final project, coursework.

Quality Management

The quality of the course is assured by the various teaching-learning methods along with a comfortable infrastructure:

- Learning material is being facilitated in a manner, that no prior programming experience is required.
- All course material is available online (including theory in a video format) through ISM e-learning platform, therefore student's learning is personalized.
- Peer-to-peer consultation with each learner is being constantly carrying out during the lectures, to ensure that students have all the support to go further.
- Collaboration among students is encouraged and facilitated to form a community where they can feel comfortable to receive help or feedback.
- All tools for the learning process are web-based and can be accessed through ISM e-platform, therefore no software purchasing/installation is required and learning can occur on any laptop or PC.
- Assessment of the problems sets' correctness and style of the code is automated, therefore students can immediately improve their code or move forward.

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 final exam and test is not permitted at all. Collaboration on the course's final project is permitted to the extent prescribed by its specification.

Reasonable	**Not 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 lectures, elsewhere, or even online, as by viewing, compiling, or running their code after you have submitted that portion of the problem set 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 final exam 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. Turning to the course's heads for help or receiving help from the course's heads during a final exam or test.	Accessing a solution to some problem prior to (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 (re-)submitting your own. 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 final exam 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.	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 final exam or midterm. Viewing another's the solution to a problem set's problem and basing your own solution on it.

Note: subject to ISM code of ethics and other ISM regulations

Course content

#	Lecture	Topics	Lecture	Seminar	Resource
1	Computational thinking & Scratch	Problem solving Inputs, Outputs Representation Unary, Binary, Decimal Abstraction ASCII, Unicode RGB Algorithms Running Time Pseudocode Scratch ● Functions, Arguments, Return Values ● Variables ● Boolean Expressions, Conditions ● Loops ● Events ● Threads	2	4	https://cs50.harvard.edu/college/2020/spring/weeks/0/ https://cs50.harvard.edu/x/2020/weeks/0/
2	Programming language	Linux Command-Line Interface Programming language	4	4	https://cs50.harvard.edu/c

		• Functions, Arguments, Return Values • Variables • Boolean Expressions, Conditions • Loops Libraries, Header Files Text Editors Terminal Windows Compiler Types Integer Overflow Floating-Point Imprecision			ollege/2020/spring/weeks1/1/ https://cs50.harvard.edu/x/2020/weeks/1/
3	Arrays	Preprocessing Compiling Assembling Linking Debugging Arrays Strings Command-Line Arguments Cryptography	4	4	https://cs50.harvard.edu/college/2020/spring/weeks/2/ https://cs50.harvard.edu/x/2020/weeks/2/
4	Algorithms	Searching • Linear Search • Binary Search Sorting • Bubble Sort • Selection Sort • Insertion Sort • Merge Sort Asymptotic Notation • O • Ω • ϴ Recursion	4	4	https://cs50.harvard.edu/college/2020/spring/weeks/3/ https://cs50.harvard.edu/x/2020/weeks/3/
5	Midterm test	Covers all the topics, which were presented before the midterm	2	2	All resources used before the test
6	Memory	• Pointers • Segmentation Faults • Dynamic Memory Allocation • Stack • Heap • Buffer Overflow • Data Structures • File I/O • Images	4	4	https://cs50.harvard.edu/college/2020/spring/weeks/4/ https://cs50.harvard.edu/x/2020/weeks/4/
7	Final project part I	Product planning Time management Consultations	2	2	-
8	Final project part II	Product application Problem solving Consultations	2	2	-

Assessment Methods

The Final Grade will be calculated as follows:

Type of an assignment	Hours	Course grade weights (%)
Problems sets (5 assignments)	56	50%
Midterm	16	15%
Final exam	16	15%
Final project	24	20%
Total:	**112**	**100%**

Note: active participation during the classes and seminars might contribute max. 1.5 point to the final evaluation.

Problem sets and the final project are evaluated along axes of correctness and style, with correctness ordinarily counting for 75% of your score and style counting for 25%.

Lateness
Late submissions (of the problem sets and the final project's milestones) will be penalized at a rate of 0.1% per minute:

- If you submit 10 minutes late, your score will be penalized 1%. Your score will thus be 99% of what it would have been if submitted on time.
- If you submit 60 minutes late, your score will be penalized 6%. Your score will thus be 94% of what it would have been if submitted on time.
- If you submit 1,000 minutes (just over 16 hours) late, your score will be penalized 100%. Your score will thus be effectively zeroed.

Regardless of submitting (problem sets and the final project) on time or 16 hours late, they have to be completed in order to be eligible for a satisfactory grade.

Midterm
Covers all the material which was presented before the midterm. Midterm form **i**s organized as 2 hours programming task. This midterm applies **open-book rules.**

Final exam
Covers all the material which was presented before and after the midterm. Final exam is organized as 2 hours programming task . This final exam applies **open-book rules and is organized during the exam session.**

Open-book rules
Students 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, which means that...

You may	You may not
browse and search the Internet (except the solution to the task), review books, review the course's own materials, use CS50 IDE or CS50 Sandbox	browse and search the Internet for the solution to the task. receive or solicit directly or indirectly any help from anyone other than the course's heads.

Take care to review the course's policy on academic honesty in its entirety. Note particularly, but not only, that

looking at another individual's work during the test is *not reasonable* and
turning to humans (besides the course's heads) for help or receiving help from humans (besides the course's heads) during the test is *not reasonable*.

Unless otherwise noted, you may call any functions we've encountered this term in the code that you write. You needn't comment code that you write, but comments may help in cases of partial credit. If having difficulty with code, you may resort to pseudocode for potential partial credit.

Among the midterm’s and final exam’s aims is to assess your newfound comfort with the course’s material and your ability to apply the course’s lessons to familiar and unfamiliar problems. And most problems aspire to teach something new.

Final project
The final project is the final assignment of the course. Students get the opportunity to choose any topic they want. So long as your project draws upon the lessons of this course, the nature of your project is entirely up to you, albeit subject to the staff's approval. Students are asked to pick the idea they want and implement with preferred technology in the way that it solves an actual problem, that you impact campus, or that change the world.

That being said, there are some provisos. You may implement your project in any programming language(s) as long as the teaching staff approves it. You are welcome to utilize any infrastructure, provided the staff ultimately has access to any hardware and software that your project requires. Turing Society has some hardware that may be used in some cases. The final project length should equivalent to at least 1.5 problem set. Final projects should be presented for the course heads in up to 3 min presentation in virtual or physical format.
Overall implementation of the final project will be taken into account when grading the final project.

Retake
In case the final grade is less than five (not passed), students can be allowed to have one retake. Retake means: retake midterm and final exam. The evaluation of the problem sets will not be affected by the retake.

Resources
Main: https://cs50.harvard.edu/college/, https://cs50.harvard.edu/x/
Optional:

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