Cypress Lake High School Center for the Arts

CS50 for AP Computer Science Principles

Teacher: Patty Gair

COURSE SYLLABUS COURSE INTRODUCTION

CS50 is Harvard University's introduction to the intellectual enterprises of computer science and the art of programming for students with a diversity of technological background and experience. CS50 for AP Computer Science Principles is an adaptation of CS50 specifically tailored to align with the AP Computer Science Principles curriculum framework. The course's assignments, materials, and resources are all identical to the version of the course taught at the college-level, albeit adapted to suit a secondary school audience. Among this course's objectives is to supply students with a comprehensive introduction to the fundamentals of the discipline of computer science. We will do so using programming in several different languages as a vehicle to introduce these fundamentals, including such topics as algorithms, abstraction, data, global impact, and internet technologies. Though the course is programming-heavy, it should be stressed that this is not a "programming course"; rather, this course should be considered one of problem-solving, creativity, and exploration. By year's end, students will have a richer understanding of the key principles of the discipline of computer science. They will be able to speak intelligently about how computers work and how they enable us to become better problem-solvers, and will hopefully be able to communicate that knowledge to others. Whether students elect to take no other computer science courses in their lifetime or consider this class the first step in a longer course of study, it is our sincere hope that students feel more comfortable with—and indeed sometimes skeptical of—the technologies that surround us each day.

PREREQUISITES

The only background required for CS50 for AP Computer Science Principles is completion of Algebra I or its equivalent.

RECOMMENDED BOOKS

No books are required for this course. However, students may wish to supplement their preparation for or review of some material with self-assigned readings relevant to the material 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. Realize that free, if not superior, resources can be found on the course's website or on the internet more generally.

For Those Less Comfortable

C Programming Absolute Beginner's Guide, 3rd Edition Greg Perry, Dean Miller Pearson Education, 2014 ISBN 0-789-75198-4 4

For Those More Comfortable

Programming in C, 4th Edition Stephen G. Kochan Pearson Education, 2015 ISBN 0-321-77641-0

The following book is recommended for those interested in understanding more about how their own computers work, for personal edification.

How Computers Work, 10th Edition Ron White Que Publishing, 2014 ISBN 0-7897-4984-X

Lastly, the following book is recommended for aspiring hackers—those interested in programming techniques and low-level optimization of code that goes beyond the scope of this course. Hacker's Delight, 2nd Edition Henry S. Warren, Jr. Pearson Education, 2013 ISBN 0-321-84268-5

PROGRAMMING ENVIRONMENTS

Several programming languages are taught in the course, and students are able to program in all of them in an environment designed specifically for the course called CS50 IDE. Students will need to sign up for a (free) account on edX (edx.org) in order to use CS50 IDE. CS50 IDE is a web-based utility (hosted on a platform known as Cloud9) with cloud storage, meaning students will be able to work on the course's programming exercises at home, school, or anywhere they have an internet connection. Instructions for setting up and using CS50 IDE are provided in the first assignment requiring its use. Additionally, students will use a drag-and-drop programming language called Scratch for some of the course's early material. Scratch is similarly a web-based environment, and it can be accessed by visiting https://scratch.mit.edu/.

ACADEMIC INTEGRITY

This course takes academic integrity quite seriously. Be assured that tools exist that make it trivially simple to detect cases of academic dishonesty, and as such this course's philosophy on academic integrity 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 endeavors to characterize both sides of that line. 5 The essence of all work that you submit to this course must be your own. Collaboration on problems is not permitted (unless explicitly stated otherwise) 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 or writing to others, but you may not view theirs. Collaboration on any quizzes and tests is not permitted at all. Collaboration on the Create Performance Task is permitted to the extent prescribed by its specifications. Below are examples that inexhaustibly 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 your instructor. If a violation of this policy is suspected and confirmed, your instructor reserves the right to impose an appropriate penalty.

Reasonable

• Communicating with classmates about 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, such 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 (as via comments in your code).

• Reviewing past years' quizzes, tests, 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 solutions to problems online so that others might help you identify and fix a bug or other issue.

• 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 problems or your own final project.

• Whiteboarding solutions to problems 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 (re-)submitting your own.

• Asking a classmate to see his or her solution to a problem before (re-)submitting your own.

• Decompiling, deobfuscating, or disassembling the sample solutions to problems available in CS50 IDE.
• Failing to cite (as with comments) the origins of code, writing, 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 when it is he or she, and not you, who is struggling to solve it.

• Looking at another individual's work during a quiz or test.

• Paying or offering to pay an individual for work that you may submit as (part of) your own. 6

• Providing or making available solutions to problems 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 problems online or elsewhere.

• Splitting a problem's workload with another individual and combining your work (unless explicitly authorized by the problem itself).

• 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.
• Using resources during a quiz beyond those explicitly allowed in the quiz's instructions.

• Viewing another's solution to a problem and basing your own solution on it.

ASSESSMENT

Because computer science is not a discipline that only lends itself to questions of right and wrong but also how and why, this course's assessment policy is designed to try to answer some or all of these questions. The course's problems are evaluated along the four axes of scope, correctness, design, and style. Scope To what extent does your submission attempt to implement the features required by the specification? Correctness To what extent is your submission consistent with our specification and free of bugs or errors? Design To what extent is your submission written well (i.e., clearly, efficiently, elegantly, and/or logically)? Style To what extent is your submission readable (i.e., code is commented and intended with aptly-named variables)? To obtain a passing grade in the course, all students must ordinarily submit all assigned problems unless otherwise granted an exception in writing by the instructor.

OVERVIEW

Consistent with the AP Computer Science Principles curriculum framework, the course's material is organized around seven so-called "big ideas" as well as six computational thinking practices. The seven big ideas are:

1. Creativity

2. Abstraction

3. Data and Information

4. Algorithms

5. Programming

6. The Internet

7. Global Impact

7 And the six computational thinking practices are:

1. Connecting Computing

2. Creating Computational Artifacts

3. Abstracting

4. Analyzing Problems and Artifacts

5. Communicating (both orally and in writing)

6. Collaborating

CORE CHAPTERS

Chapter 0: Computers and Computing (Completion Time: 3 weeks) In this chapter, students will learn about computer hardware and architecture, the language of computers – binary, how information is encoded so that humans can understand it, and begin to explore the ways in which computers process

information. Topics within this Chapter 0 Computers and Computing 1 How Computers Work 2 Bits and Bytes 3 Hardware 4 Memory 5 Binary Numbers 6 ASCII 7 Algorithms

Chapter 1: Building Blocks of Programming (Completion Time: 4 weeks) In this chapter, students will learn the fundamentals of computer programming, to permit them to begin to manipulate information and data and command a computer to do calculations they wish for it to perform. Topics within this Chapter 0 Pseudocode 1 Scratch 2 Syntax 3 Variables 4 Data Types 5 Operators 6 Boolean Expressions and Conditionals 7 Loops

Chapter 2: Putting the Blocks Together (Completion Time: 5 weeks) In this chapter, students will expand upon their knowledge of the fundamentals of computer programming, and begin building abstractions of their own, understanding how to represent 11 collections of data and write functions/methods/subroutines to allow their code to be more portable. Topics within this Chapter 0 Compiling 1 Functions 2 Arrays and Strings 3 Command-Line Interactions 4 Exit Codes 5 Libraries 6 Typecasting 7 Bugs and Debugging

Chapter 3: Thinking Computationally (Completion Time: 4 weeks) In this chapter, students begin to consider different algorithms and how they process information, appreciating the differences between algorithms and gaining the vocabulary necessary to speak 13 on their relative merits and disadvantages. They will also be introduced to the concept of undecidability, the fact that computers are not capable of answering every question we ask them. Topics within this Chapter 0 Linear Search 1 Bubble Sort 2 Selection Sort 3 Insertion Sort 4 Binary Search 5 Time Complexity 6 Unsolvable Problems 7 Simulation

Chapter 4: Design, Elegance, and Efficiency (Completion Time: 4 weeks) As students begin to wrap up their introduction to C (except for those in the Fast-Track Chapters, below), they are challenged to consider questions as to what makes a solution good, and what makes a program beautiful. In this chapter in particular, we challenge students to consider not just right and wrong, but how and why, discussing design trade-offs and why sometimes indeed the discipline of computer science boils down to determining what trade-off is appropriate under a given set of circumstances. Topics within this Chapter 0 Principles of Good Design 1 ncurses 2 Structures and Encapsulation 3 Recursion 4 Merge Sort 5 Hexadecimal 6 File I/O 7 Images 8 Version Control

Performance Task 1: Explore – Impact of Computing Innovations (Completion Time: 2 weeks) After completing Chapter 4, students will start the through-course assessment Explore—Impact of Computing Innovations. Students are allocated eight hours of in-class time for this exercise.– Students are provided the required amount of class time to complete the AP Through Course Assessment Explore – Impact of Computing Innovations Performance Task.

Chapter 5: Networking and the Internet (Completion Time: 3 weeks) At this point in the course, we transition from programming in a mostly command-line environment to taking our applications to scale via the Internet. First, however, students are introduced to the technologies underpinning this thing we know as “the Internet” before beginning to explore web programming by building simple pages of their own and making them accessible to the world via CS50 IDE. Topics within this Chapter 0 Internet Basics 1 IP Addresses 2 DNS and DHCP 3 Routers 4 TCP and IP 5 HTTP 6 Trust Models and Open Source 7 Cybersecurity 8 HTML 9 CSS

Chapter 6: Problem Solving in an Interconnected World (Completion Time: 5 weeks) 19 Lastly, students build upon their knowledge gained in the course to learn several new programming languages with abstractions built in that allow them to go far beyond what simply C and Scratch are able to do. They solve more complex problems that require processing large amounts of data and dealing with processes that scale, and see how these techniques can be applied to confront the challenges computer scientists will be contending with in the future. Topics within this Chapter 0 PHP 1 PHP for Web Programming 2 SQL 3 MVC 4 JavaScript 5 Ajax 6 Artificial Intelligence 7 Virtual and Augmented Reality

Performance Task 2: Create – Applications from Ideas (Completion Time: 3 weeks) After competing Chapter 6, students complete the through-course assessment Create—Applications from Ideas. Students are allocated twelve hours of in-class time for this exercise.– Students are provided the required amount of class time to complete the AP Through Course Assessment Create – Applications from Ideas Performance Task. 21

FAST-TRACK CHAPTERS For students who find themselves acclimating well to the course's material, have prior background in computer science, or consider themselves among those more comfortable, the course provides a fast-track that allows these students to explore material related to that covered in the course's core chapters. The material in these chapters goes beyond the scope of the AP Computer Science Principles curriculum framework, but provides a venue for students to gain a deeper appreciation for the computational power of lower-level programming languages and an opportunity to explore data structures and data management techniques. These fast-track chapters are designed to come, in the course's narrative, between Chapters 4 and 5. Because these chapters are strictly optional, no completion timeline is provided and no mapping to the AP Computer Science Principles curriculum framework is given.

Fast-Track Chapter A: Managing Data In this fast-track chapter, students learn about different ways of managing sets of data, both for sets that will rapidly and boundlessly grow, and also for easier subsequent indexing and searching. Visit https://ap.cs50.net/curriculum/A/ to view content and descriptions of the following: Topics within this Chapter 0 Stacks 1 Queues 2 Pointers 3 Dynamic Memory 4 Valgrind 5 Trees 6 Tries 7 Linked Lists 8 Hash Tables

Fast-Track Chapter B: Developer Toolbox 22 In this fast-track chapter, students will complete their introduction to C by diving into some of its most complex syntax, and implement strategies for data compression, abstractions, API development, and interfacing for web-traffic using C. Visit https://ap.cs50.net/curriculum/B/ to view content and descriptions of the following: Topics within this Chapter 0 Abstraction and API 1 Data Compression 2 Huffman Coding 3 Scalability 4 Collaboration