Teaching Tomorrow's Technology.
CompuScholar

Computer Science Foundations

For AP Computer Science Principles

Course Syllabus and Planner (Updated Fall 2023)

Course Overview

This AP Computer Science Principles (CSP) class uses the **CompuScholar Computer Science Foundations[1]** curriculum as the primary resource. It is taught as a one-year (two-semester) sequence and covers all required topics in the "Computer Science Principles" Course Description published by the College Board. The Python language is taught as the basis for programming topics.

Students need to have typical computer usage skills prior to starting this course; other introductory programming courses are not required. All required concepts are taught from the ground up in a fun, step-by-step manner. The course includes uses a variety of multimedia content such as full-color, interactive text, narrated instructional videos, and guided exercises. Strong emphasis is placed on hands-on programming labs to demonstrate mastery of lesson concepts.

The **CompuScholar Computer Science Foundations** curriculum is **fully aligned with the AP Computer Science Principles Course and Exam Description**. This allows teachers to easily leverage the additional material and practice questions in the AP Classroom.

College-Level Textbook Resource

This course uses the following electronic textbook:

[1] **Computer Science Foundations** online text, by CompuScholar, Inc. 2023, ISBN 978-1-946113-02-3. https://www.compuscholar.com/schools/courses/csfoundations/

Course Material

The course material is designed to appeal to a variety of students, from traditional learners who thrive on written text to audio-visual students who enjoy a multimedia format. All content is delivered through an online system that allows students to work seamlessly both in the classroom and at home.

The course consists of the following student-facing elements:

- **Instructional Videos** – optional (not required), but enjoyed by many students as an audio-visual introduction and reinforcement of the lesson topics.
- **Lesson Text** – required reading, contains full topic details and live coding exercises
- **Quizzes and Exams** – multiple-choice and automatically graded by our system
- **Chapter Activities** – hands-on projects, submitted for a grade

Teachers additionally have access to:

- **Teacher's Guides** – for each lesson, with suggested classroom discussion questions
- **Quiz and Exam Answer Keys** – PDFs for quick reference
- **Activity Solution Guides** – fully coded activity solutions for each chapter activity

Programming Environment and Device Requirements

CompuScholar provides an in-browser Python coding environment. This online feature may be used by students to complete all exercises and activities in all required AP chapters. When using the online coding environment:

- **No local software installation is needed to prepare for the AP exam.**
- **The AP material can be completed from any web browser on any device (including Chromebooks and tablets).**

Later, optional chapters contain a mixture of activities. AP CSP teachers may select any of these topics for students after the AP exam. Some optional activities can be done in CompuScholar's online environment, while others are completed offline on a local computer.

Project Grading

Each chapter normally contains one or more hands-on, graded activities. The Python coding activities **are auto-graded by the CompuScholar system**. Teachers have complete control over the auto-graded results.

Some activities in other chapters, especially those not involving a programming project, result in creative student work or work that is completed offline. The teacher is responsible for grading those creative or external projects.

Course Navigation

Chapters 1 – 22 should be completed in sequence and cover all required topics on the AP CSP exam, plus certain other highly recommended software skills. These chapters include substantial, hands-on lab work and guidance on the CSP "performance task" and exam preparation.

Typical classes will finish all required AP CSP content prior to the exam administration in May. We recommend using the remaining time before the exam to review the College Board's published practice exams and any other external source of practice problems. Students should also be given sufficient time to complete the performance task.

Chapters 23 – 29 contain optional topics that are not required for AP CSP. Teachers may review and select any of these optional topics for students as time permits after the AP CSP exam.

Supplemental Chapters contain a variety of enrichment topics that may be required by individual states to satisfy requirements for other coding or digital literacy courses.

"Big Ideas" and Sample Student Activities

The hands-on lessons and student activities found within each chapter reinforce one or more "**Big Ideas**" from the AP CSP Course and Exam Description (CED). The table below provides more detail on **one** example activity for each Big Idea.

Big Idea 1: Creative Development (CRD)
Chapter 14: Collaborative Design and Development **Activity: Collaborative Project** Students will form small teams and complete a small SDLC with traditional stages. The teams will first brainstorm a problem and solution, then create requirements and design documents. Team members will all participate in the implementation (coding) and testing phases. Feedback from the class or teacher is incorporated after each step.
Big Idea 2: Data (DAT)
Chapter 17: Understanding Data **Activity: Census Analysis** Students will use online spreadsheet tools to analyze U.S. Census data from 2000, 2010, and 2020. The analysis includes cleaning data, sorting, and filtering, identification of meta-data, searching. Line and pie charts are used to draw conclusions about changes in population over time and the relative sizes of U.S. states based on population.
Big Idea 3: Algorithms and Programming (AAP)
Chapter 16: Real-World Algorithms **Activity: Knapsack Problem** The Knapsack problem is a classic challenge that is generally undecidable with larger number of options. Students will implement two different heuristic algorithms to provide answers for randomized knapsack challenges, then compare the performance of each heuristic to select the best option.
Big Idea 4: Computer Systems and Networks (CSN)
Chapter 2: Networking **Activity: Network Analysis** Students will complete a series of 3 network-related challenges, including vocabulary identification, connectivity and failure point analysis, and identification of alternate packet routes when the ideal path is blocked due to hub or channel failures.
Big Idea 5: Impact of Computing (IOC)
Chapter 19: Legal and Ethical Concerns **Activity: Ethics Illustration** Students will review the Association for Computing Machinery (ACM) Code of Ethics, select one point of interest, and research an example where someone has violated that ethical principle. Students will create and share a poster illustrating the violation, the group(s) targeted, the impact, the laws violated, and the legal penalties that should be applied to the selected violation.

In the later "**Course Planner**" section, every chapter is also labeled with the Big Ideas (CRD, DAT, AAP, CSN, IOC) relevant for the lessons and activities in that chapter.

"Computational Thinking Practices" and Student Activities

The hands-on lessons and student activities found within each chapter reinforce one or more "**Computational Thinking Practices**" (**CTP**) from the AP CSP Course and Exam Description (CED). The table below describes **one** example for each CTP.

CTP 1: Computational Solution Design
Chapter 13: Mid-Term Project Students will design and complete a small card game ("GOPS" - "Game of Perfect Strategy") over a series of guided activities. **Activity #1: "GOPS" Part 1** Students will study starting code and the program layout, then design and implement the main program logic to control the overall game flow. **Activity #2: "GOPS" Part 2** Students will implement several methods with parameters, including lists, to do things like shuffle the card deck, have the computer AI opponent select a card to play, and determine the game winner. **Activity #3: "GOPS" Analysis** Students will reflect on their completed project, answering questions similar in scope to the CTP Written Responses they will be required to complete for their own CPT project as part of the AP CSP exam.
CTP 2: Algorithms and Program Development
Chapter 15: Designing Algorithms **Activity: Cat and Mouse** Students will use flowcharts and pseudocode to design their own algorithm to make a cat "chase" a mouse on a small grid. Both cat and mouse are represented by an X, Y coordinate pair on a grid, and the student algorithm must evaluate all possible moves for the cat and select the move that will take it closest to the mouse.
CTP 3: Abstraction in Program Development
Chapter 9: Lists **Activity: Burger Castle** Students will implement a menu ordering system using lists to represent menu choices, item descriptions, and the customer's order. Abstraction through variables and lists is used to manage program complexity, including display of the menu and recording and summarizing the customer's order.

CTP 4: Code Analysis
Chapter 7: Finding and Fixing Problems **Activity: Chat-Bot** Students are provided with a set of program requirements and a "complete" program that has numerous bugs. Students will study the starting code to identify the purpose of each section, find, and fix each bug. Numerous test cases are provided to help students verify the expected output.
CTP 5: Computing Innovations
Chapter 18: Impacts of Computing **Activity: Impact Study** Students are provided a sample list of computing innovations (e.g., self-driving cars, smart assistants, social media, rideshares, home video monitoring) and will select one topic of interest. Students will do online research to summarize the technology, identify the positive and harmful effects, and understand the groups of people impacted by the positive and harmful effects. Students will also review security and privacy concerns arising from the innovation.
CTP 6: Responsible Computing
Chapter 20: Cybersecurity **Activity: Stuxnet Worm** Students will perform a case study of the Stuxnet Worm. As part of their study, students will learn how multiple nations likely collaborated on the worm, how systems were targeted, and how supposedly secure computing devices were infected.

In the later "**Course Planner**" section, every chapter is also labeled with the CTP relevant for the lessons and activities in that chapter.

Computing Innovations

Computing innovations are studied in multiple areas. The table below describes three exercises (**Innovation 1, 2, 3**) and how students will participate (**Prompt A** - Explore beneficial and harmful effects, **Prompt B** - Identify how data is consumed, produced, or transformed, or **Prompt C** - Identify data privacy, security, or storage concerns).

Computing Innovation 1, Prompt B
Chapter 18, Lesson 2 Exercise: Innovation Study The class will discuss how GPS mapping systems work, in particular the data needed to support ideal route mapping (street networks, real-time traffic, current locations, etc.) How is each type of data obtained and stored online? How is that data combined to produce the final route recommended to drivers?
Computing Innovation 2, Prompt A
Chapter 18 Activity: Impact Study Students are provided a sample list of computing innovations (e.g., self-driving cars, smart assistants, social media, rideshares, home video monitoring) and will select one topic of interest. Students will do online research to summarize the technology, identify the positive and harmful effects, and understand the groups of people impacted by the positive and harmful effects. Students will also review security and privacy concerns arising from the innovation.
Computing Innovation 3, Prompt C
Chapter 20, Lesson 2 Exercise: Home Video Monitoring Systems In small groups or individually, students will investigate how home video monitoring systems (e.g. "Ring") transmit and store data. They will identify privacy and security concerns for home video data stored in a cloud and subject to legal requisition or illegal access.

Course Planner

The following pages contain an outline of course content and typical pacing. Additional, detailed mappings to AP CSP Learning Objectives and Essential Knowledge (LOEKs) are appended at the end of this document.

A typical school year consists of 36 calendar weeks or 180 days of school. After completing the first 22 chapters, most classes will have several weeks left for AP exam prep, performance tasks, make-up work, and optional topics. Teachers can select from optional topics before and after the exam, as time permits.

Each "day" listed below represents one typical class period of 45 – 60 minutes. In most cases, students will complete one lesson per day (including the quiz), 1 day per lab, and 1 day per chapter test. Some classes may move faster or slower than the suggested pace.

Semester 1 Timeline

Days	CompuScholar Chapter and Lab	Big Idea and Computational Thinking Practice (CTP)
5	**Chapter 1: Computing Concepts** * Evolution of Computers * Computer Hardware * Computer Software **LAB: Using Peripherals** (Online, teacher-graded)	**Big Idea(s): CSN** **CTP(s): 1.A, 1.D** Students will investigate historical computing milestones, count and classify the computing devices in their home, discuss the software they use every day, and research technical information about one peripheral connected to a computer.
6	**Chapter 2: Networking** * Network Hardware * How the Internet Works * Internet Scalability & Fault Tolerance * Parallel and Distributed Computing **LAB: Network Analysis**	**Big Idea(s): CSN** **CTP(s): 1.A, 1.D** Students will survey the network connections in their home, identify key network terms, predict the impact from failure of key network components, and identify alternate packet routes to bypass failures.

Days	CompuScholar Chapter and Lab	Big Idea and Computational Thinking Practice (CTP)
5	**Chapter 3: Fundamentals of Python** * Introduction to Python * Running Python Programs * Writing Python Code **LAB: Class Schedule** (Online, auto-graded)	**Big Idea(s): AAP, CRD** **CTP(s): 1.B, 2.B, 4.A** Students will explore Python documentation, learn to run Python programs, and begin writing basic code with sequential output statements.
6	**Chapter 4: Working with Data** * Introduction to Abstraction * Data Types and Variables * Using Numeric Variables * Using String Variables **LAB: Cash Register** (Online, auto-graded)	**Big Idea(s): AAP, DAT** **CTP(s): 2.B, 3.A, 3.B, 3.C, 4.A** Students will discuss abstraction concepts, identify ideal data types, learn and apply basic mathematical expressions and assignments, and represent text as strings.
5	**Chapter 5: Input and Output** * Printing with Parameters * Getting Input from a User * String Formatting **LAB: Character Art** (Online, auto-graded)	**Big Idea(s): AAP, CRD** **CTP(s): 2.B, 3.A,** Students will learn to format and display strings, get input from users, and create interactive programs.
6	**Chapter 6: Making Decisions** * Logical Expressions * The "if" Statement * Logical Operators * More Complex Expressions **LAB: Blue Moon** (Online, auto-graded)	**Big Idea(s): AAP** **CTP(s): 2.B, 4.A, 4.B** Students will evaluate simple and complex Boolean expressions and create a program incorporating decision chains ("if/else-if/else")
5	**Chapter 7: Finding and Fixing Problems** * Types of Errors * Troubleshooting Tools * Using the Python Debugger **LAB: Chat-Bot** (Online, auto-graded)	**Big Idea(s): CRD** **CTP(s): 4.A, 4.B, 4.C** Students will learn to identify and troubleshoot problems using a variety of tools, including the Python debugger. Students will correct problems in a provided program to ensure proper operation.

Days	CompuScholar Chapter and Lab	Big Idea and Computational Thinking Practice (CTP)
5	**Chapter 8: Loops** * "For" Loops * "While" Loops * Break, Continue, and Else **LAB: Vowel Eraser** (Online, auto-graded)	**Big Idea(s): AAP** **CTP(s): 1.D, 2.B, 4.B** Students will implement algorithms using "for" and "while" loops, including nested loops and programs with interactive user input.
6	**Chapter 9: Lists** * Lists and Tuples * List Functions * List Traversal * Sequential and Binary Searches **LAB: Burger Castle** (Online, auto-graded)	**Big Idea(s): AAP** **CTP(s): 3.B, 3.C, 4.C** Students will write code to manipulate lists in the context of a larger program, including initialization, list functions, traversal, and searching.
6	**Chapter 10: Math Concepts** * Libraries and Documentation * Random Numbers * The Math Library * Computer Number Systems **LAB: Math Contest** (Online, auto-graded)	**Big Idea(s): AAP, DAT** **CTP(s): 1.C, 2.B, 3.C** Students will learn about reusable libraries and explore online Python API reference material. Students will learn to use the random and math libraries in useful programs. Students will also study the binary number system and conversions to decimal.
5	**Chapter 11: Working with Strings** * Character Data * String Functions * Input Validation (try/except) **LAB: Pig Latin Translator** (Online, auto-graded)	**Big Idea(s): AAP** **CTP(s): 3.A, 3.B, 4.B, 4.C** Students will learn to extract characters from strings and manipulate strings with a variety of string methods. They will write code to verify user input and encode / decode data using a defined encoding scheme.

Days	CompuScholar Chapter and Lab	Big Idea and Computational Thinking Practice (CTP)
6	**Chapter 12: Creating Functions** * Writing and Calling Functions * Function Inputs and Outputs * Local and Global Scope * Procedural Abstraction **LAB: Verification Function** (Online, auto-graded)	**Big Idea(s): AAP** **CTP(s): 1.B, 1.D, 2.B, 3.B, 3.C, 4.A** Students will learn to write their own functions with parameters and return data. Students will learn to identify blocks of code that can be extracted as functions and applied to future programs.
7	**Chapter 13: Mid-Term Project** * Introducing "GOPS" **LAB 1: "GOPS" Part 1** (Online, auto-graded) **LAB 2: "GOPS" Part 2** (Online, auto-graded) **LAB 3: "GOPS" Analysis** (Online, teacher-graded)	**Big Idea(s): CRD, AAP** **CTP(s): 1.A, 2.B, 4.B** The mid-term project leads students through development of a specific program that is comparable, in scope, to the Create Performance Task. Students will gain experience writing a moderately complex program in small stages.
73	**Total Days, Semester 1**	

Semester 2 Timeline

Days	CompuScholar Chapter and Lab	Big Idea and Computational Thinking Practice (CTP)
7	**Chapter 14: Collaborative Design and Development** * Collaboration * Design Processes and Teamwork * Requirements and Design Documents * Testing Your Code **LAB 1: Project Requirements and Design** **LAB 2: Project Implementation** **LAB 3: Project Testing** (All Online, teacher-graded)	**Big Idea(s): CRD** **CTP(s): 1.C, 4.C, 6.A, 6.B, 6.C** Students will collaborate to produce a unique, creative project using a basic software design lifecycle (SDLC). Activities include group production of requirements and design documents, test plans, and program code.

Days	CompuScholar Chapter and Lab	Big Idea and Computational Thinking Practice (CTP)
5	**Chapter 15: Designing Algorithms** * Designing with Flowcharts * Writing Pseudocode * Common Math Algorithms **LAB: Cat and Mouse** (Online, auto-graded)	**Big Idea(s): AAP** **CTP(s): 1.B, 2.A, 2.B, 4.A, 4.B** Students will develop an abstract algorithm using flowcharts and pseudocode. They will also study common mathematical algorithms using these tools, and implement their own algorithm given certain program requirements.
5	**Chapter 16: Real-World Algorithms** * Algorithm Efficiency * Undecidable Problems * Simulations **LAB: Knapsack Problem** (Online, auto-graded)	**Big Idea(s): AAP** **CTP(s): 1.B, 2.A, 2.B, 4.B** Students will compare the efficiency of known algorithms, classify algorithms as decidable or undecidable, explore algorithms that simulate real-world processes, and apply heuristics to find the best solution.
6	**Chapter 17: Understanding Data** * Extracting Info from Data * Challenges with Processing Data * Using Data in Programs * Data Compression **LAB: Census Analysis** (Online, teacher-graded)	**Big Idea(s): DAT** **CTP(s): 1.A, 5.B, 5.D** Students will learn to apply spreadsheet tools for data analysis, clean data from users, and work with data compression schemes. Analysis will be performed on real-world data to draw conclusions.
5	**Chapter 18: Impact of Computing** * Digital Divide and Bias * Effects of Computing Innovations * Crowdsourcing and Citizen Science **LAB: Impact Study** (Online, teacher-graded)	**Big Idea(s): IOC** **CTP(s): 1.A, 5.A, 5.C, 5.E, 6.B** Students will study regional Internet access, explore a variety of computing innovations, study well-known crowdsourcing and citizen science projects and resources, and research and analyze the impact of computing innovations.

Days	CompuScholar Chapter and Lab	Big Idea and Computational Thinking Practice (CTP)
5	**Chapter 19: Legal and Ethical Concerns** * Computing Ethics * Computing Laws * Intellectual Property **LAB: Ethics Illustration** (Online, teacher-graded)	**Big Idea(s): IOC** **CTP(s): 1.A, 5.E, 6.C** Students will review and discuss an AUP as a class, explore software licenses and intellectual property rules, legal consequences of computer crimes, and research, illustrate and share the impact of and action violating a code of ethics.
5	**Chapter 20: Cybersecurity** * Protect Yourself Online * Computer Security * Social Engineering **LAB: Stuxnet Worm** (Online, auto-graded)	**Big Idea(s): IOC** **CTP(s): 1.A, 5.A, 5.D, 5.E, 6.B** Students will review and implement personal security guidelines, discuss experiences with social engineering, and perform a case study of the Stuxnet Worm.
4	**Chapter 21: Preparing for the CSP Exam** * Exam Format and Scoring * Exam Reference Sheet and Pseudocode * Robot Problems * AP Classroom Resources	Students will examine the CSP exam format and scoring, explore the reference sheet and review pseudocode standards, practice creating robot algorithms, and learn how to access AP Classroom resources.
16+	**Chapter 22: CSP Performance Task** * Introduction to the Create Performance Task * Program Code * Video and Project Reference * Written Responses **Lab: Performance Task Development** (Online, No grading)	Students will learn about requirements for the CPT, review the required written responses, and implement their own CPT. **Students will be provided 12+ class hours for Performance Task Development**
58	**Total Days, Semester 2 (all required AP CSP topics complete at this point)**	

Classes who complete the first 22 chapters at this point have spent approximately 132 days and covered all tested AP CSP topics. **The remaining class time should be spent in preparation for the AP exam and covering other optional topics as desired.**

The following tables suggest the timeline needed for each **optional** or **supplemental chapter**, along with notes as to the programming environment and grading approach.

Optional WEB DESIGN Unit

Days	CompuScholar Chapter and Lab	Notes
5	**Chapter 23: Creating Web Pages** * Getting Started with HTML * Creating HTML Files * HTML File Layout **LAB: Beginning Web Page**	**WEB DESIGN UNIT** Offline, teacher-graded
5	**Chapter 24: Web Page Design** * Body Elements * Using Colors * Design Rules **LAB: Formatted Web Page**	**WEB DESIGN UNIT** Offline, teacher-graded
6	**Chapter 25: Links, Images and Animation** * Adding Hyperlinks * Using Images * Adding Animation * Simple JavaScript **LAB: Final Website**	**WEB DESIGN UNIT** Offline, teacher-graded
16	**Days in the Web Design Unit**	

Optional COMPUTER SKILLS Unit

Days	CompuScholar Chapter and Lab	Notes
5	**Chapter 26: Operating Systems** * Popular Operating Systems * Managing Your OS * Managing Your Applications **LAB: OS Report**	**COMPUTER SKILLS UNIT** Online, teacher-graded
5	**Chapter 27: Computer Files** * Understanding Files and Folders * Managing Files on Your Computer * File Associations **LAB: Mystery Files**	**COMPUTER SKILLS UNIT** Online, teacher-graded
5	**Chapter 28: Search Engines** * Using Search Engines * Search Results * Verifying and Citing Sources **LAB: Search Report**	**COMPUTER SKILLS UNIT** Online, teacher-graded
15	**Days in the Computer Skills Unit**	

Optional COMPUTER CAREERS Unit

Days	CompuScholar Chapter and Lab	Notes
6	**Chapter 29: Computer Careers** * Computer Career Opportunities * Professionalism in the Workplace * Workplace Safety * Student Organizations **LAB: Exploring Computing Careers**	**COMPUTER CAREERS UNIT** Offline, teacher-graded
6	**Days in the Computer Careers Unit**	

Optional Supplemental Chapters and Topics

Days	CompuScholar Chapter and Lab	Notes
3	**Supplemental Chapter 1: Python on Your Computer** * Installing Python * Managing Project Directories * Python IDLE Development Environment	Offline
3	**Supplemental Chapter 2: File I/O with Python** * Text File Reading and Writing * Managing Files with the OS Module * File Resource and Error Handling	Online
6	**Days in the Supplemental Chapters**	

The following pages contain detailed cross-reference tables that map every AP Computer Science Principles topic and essential knowledge to specific course chapters and lessons. For convenience, these cross-references are also available as a separate document at the following online location:

https://www.compuscholar.com/docs/apcsp/AP_CSP_Topic_Cross_Reference.pdf

CompuScholar, Inc.

Alignment to the College Board AP **Computer Science Principles**

Learning Objectives and Essential Knowledge (LOEK)

AP Course Details:

Course Title:	AP Computer Science Principles
Grade Level:	9th - 12th grades
Standards Version:	Fall 2023
Standards Link:	AP CSP Exam Description Page

CompuScholar Course Details:

Course Title:	Computer Science Foundations
Course ISBN:	978-1-946113-02-3
Course Year:	2023

Note 1: Citation(s) listed may represent a subset of the instances where objectives are met throughout the course.

Note 2: Citation(s) for a "Lesson" refer to the "Lesson Text" elements and associated "Activities" within the course, unless otherwise noted. The "Instructional Video" components are supplements designed to introduce or reinforce the main lesson concepts, and the Lesson Text contains full details.

AP CSP Course Description

This course teaches students introductory computer science concepts, including all required topics defined by the College Board's AP Computer Science Principles course description.

AP CSP Big Ideas and Learning Objectives

BIG IDEA 1: CREATIVE DEVELOPMENT	**CITATION(S)**
TOPIC 1.1 Collaboration	
CRD-1.A.1 - A computing innovation includes a program as an integral part of its function.	Chapter 14, Lesson 1
CRD-1.A.2 - A computing innovation can be physical (e.g., self-driving car), nonphysical computing software (e.g., picture editing software), or a nonphysical computing concept (e.g., e-commerce).	Chapter 14, Lesson 1
CRD-1.A.3 - Effective collaboration produces a computing innovation that reflects the diversity of talents and perspectives of those who designed it.	Chapter 14, Lesson 1
CRD-1.A.4 - Collaboration that includes diverse perspectives helps avoid bias in the development of computing innovations.	Chapter 14, Lesson 1
CRD-1.A.5 - Consultation and communication with users are important aspects of the development of computing innovations.	Chapter 14, Lesson 1

CRD-1.A.6 - Information gathered from potential users can be used to understand the purpose of a program from diverse perspectives and to develop a program that fully incorporates these perspectives.	Chapter 14, Lesson 1
CRD-1.B.1 - Online tools support collaboration by allowing programmers to share and provide feedback on ideas and documents.	Chapter 14, Lesson 1
CRD-1.B.2 - Common models such as pair programming exist to facilitate collaboration.	Chapter 14, Lesson 1
CRD-1.C.1 - Effective collaborative teams practice interpersonal skills, including but not limited to: * communication * consensus building * conflict resolution * negotiation	Chapter 14, Lesson 1
TOPIC 1.2: Program Function and Purpose	
CRD-2.A.1 - The purpose of computing innovations is to solve problems or to pursue interests through creative expression.	Chapter 18, Lesson 2
CRD-2.A.2 - An understanding of the purpose of a computing innovation provides developers with an improved ability to develop that computing innovation.	Chapter 18, Lesson 2
CRD-2.B.1 - A program is a collection of program statements that performs a specific task when run by a computer. A program is often referred to as software.	Chapter 1, Lesson 3 Chapter 3, Lesson 3
CRD-2.B.2 - A code segment is a collection of program statements that is part of a program.	Chapter 3, Lesson 3
CRD-2.B.3 - A program needs to work for a variety of inputs and situations.	Chapter 3, Lesson 3
CRD-2.B.4 - The behavior of a program is how a program functions during execution and is often described by how a user interacts with it.	Chapter 3, Lesson 3
CRD-2.B.5 - A program can be described broadly by what it does, or in more detail by both what the program does and how the program statements accomplish this function.	Chapter 3, Lesson 3
CRD-2.C.1 - Program inputs are data sent to a computer for processing by a program. Input can come in a variety of forms, such as tactile, audio, visual, or text.	Chapter 3, Lesson 3 Chapter 5, Lesson 2
CRD-2.C.2 - An event is associated with an action and supplies input data to a program.	Chapter 5, Lesson 2
CRD-2.C.3 - Events can be generated when a key is pressed, a mouse is clicked, a program is started, or any other defined action occurs that affects the flow of execution.	Chapter 5, Lesson 2
CRD-2.C.4 - Inputs usually affect the output produced by a program.	Chapter 3, Lesson 3 Chapter 5, Lesson 2
CRD-2.C.5 - In event-driven programming, program statements are executed when triggered rather than through the sequential flow of control.	Chapter 5, Lesson 2

CRD-2.C.6 - Input can come from a user or other programs.	Chapter 3, Lesson 3 Chapter 5, Lesson 2
CRD-2.D.1 - Program outputs are any data sent from a program to a device. Program output can come in a variety of forms, such as tactile, audio, visual, or text.	Chapter 5, Lesson 1
CRD-2.D.2 - Program output is usually based on a program's input or prior state (e.g., internal values).	Chapter 5, Lesson 1
TOPIC 1.3: Program Design and Development	
CRD-2.E.1 - A development process can be ordered and intentional, or exploratory in nature.	Chapter 14, Lesson 2
CRD-2.E.2 - There are multiple development processes. The following phases are commonly used when developing a program: * investigating and reflecting * designing * prototyping * testing	Chapter 14, Lesson 2
CRD-2.E.3 - A development process that is iterative requires refinement and revision based on feedback, testing, or reflection throughout the process. This may require revisiting earlier phases of the process.	Chapter 14, Lesson 2
CRD-2.E.4 - A development process that is incremental is one that breaks the problem into smaller pieces and makes sure each piece works before adding it to the whole.	Chapter 14, Lesson 2
CRD-2.F.1 - The design of a program incorporates investigation to determine its requirements.	Chapter 14, Lesson 3
CRD-2.F.2 - Investigation in a development process is useful for understanding and identifying the program constraints, as well as the concerns and interests of the people who will use the program.	Chapter 14, Lesson 3
CRD-2.F.3 - Some ways investigation can be performed are as follows: * collecting data through surveys * user testing * interviews * direct observations	Chapter 14, Lesson 3
CRD-2.F.4 - Program requirements describe how a program functions and may include a description of user interactions that a program must provide.	Chapter 14, Lesson 3
CRD-2.F.5 - A program's specification defines the requirements for the program.	Chapter 14, Lesson 3
CRD-2.F.6 - In a development process, the design phase outlines how to accomplish a given program specification.	Chapter 14, Lesson 2
CRD-2.F.7 - The design phase of a program may include: * planning and storyboarding * organizing the program into modules and functional components * creation of diagrams that represent the layouts of the user interface * development of a testing strategy for the program	Chapter 14, Lesson 2

CRD-2.G.1 - Program documentation is a written description of the function of a code segment, event, procedure, or program and how it was developed.	Chapter 14, Lesson 3
CRD-2.G.2 - Comments are a form of program documentation written into the program to be read by people and do not affect how a program runs.	Chapter 14, Lesson 3
CRD-2.G.3 - Programmers should document a program throughout its development.	Chapter 14, Lesson 3
CRD-2.G.4 - Program documentation helps in developing and maintaining correct programs when working individually or in collaborative programming environments.	Chapter 14, Lesson 3
CRD-2.G.5 - Not all programming environments support comments, so other methods of documentation may be required.	Chapter 14, Lesson 3
CRD-2.H.1 - It is important to acknowledge any code segments that were developed collaboratively or by another source.	Chapter 14, Lesson 3
CRD-2.H.2 - Acknowledgement of a code segment(s) written by someone else and used in a program can be in the program documentation. The acknowledgement should include the origin or original author's name.	Chapter 14, Lesson 3
TOPIC 1.4: Identifying and Correcting Errors	
CRD-2.I.1 - A logic error is a mistake in the algorithm or program that causes it to behave incorrectly or unexpectedly.	Chapter 7, Lesson 1
CRD-2.I.2 - A syntax error is a mistake in the program where the rules of the programming language are not followed.	Chapter 7, Lesson 1
CRD-2.I.3 - A run-time error is a mistake in the program that occurs during the execution of a program. Programming languages define their own runtime errors.	Chapter 7, Lesson 1
CRD-2.I.4 - An overflow error is an error that occurs when a computer attempts to handle a number that is outside of the defined range of values.	Chapter 7, Lesson 1
CRD-2.I.5 - The following are effective ways to find and correct errors: * test cases * hand tracing * visualizations * debuggers * adding extra output statement(s)	Chapter 7, Lesson 2 Chapter 7, Lesson 3 Chapter 14, Lesson 4
CRD-2.J.1 - In the development process, testing uses defined inputs to ensure that an algorithm or program is producing the expected outcomes. Programmers use the results from testing to revise their algorithms or programs.	Chapter 7, Lesson 2 Chapter 14, Lesson 4
CRD-2.J.2 - Defined inputs used to test a program should demonstrate the different expected outcomes that are at or just beyond the extremes (minimum and maximum) of input data.	Chapter 7, Lesson 2 Chapter 14, Lesson 4
CRD-2.J.3 - Program requirements are needed to identify appropriate defined inputs for testing.	Chapter 7, Lesson 2 Chapter 14, Lesson 4

BIG IDEA 2: DATA	CITATION(S)
TOPIC 2.1: Binary Numbers	
DAT-1.A.1 - Data values can be stored in variables, lists of items, or standalone constants and can be passed as input to (or output from) procedures.	Chapter 4, Lesson 1
DAT-1.A.2 - Computing devices represent data digitally, meaning that the lowest-level components of any value are bits.	Chapter 10, Lesson 4
DAT-1.A.3 - Bit is shorthand for binary digit and is either 0 or 1.	Chapter 10, Lesson 4
DAT-1.A.4 - A byte is 8 bits.	Chapter 10, Lesson 4
DAT-1.A.5 - Abstraction is the process of reducing complexity by focusing on the main idea. By hiding details irrelevant to the question at hand and bringing together related and useful details, abstraction reduces complexity and allows one to focus on the idea.	Chapter 4, Lesson 1 Chapter 10, Lesson 4
DAT-1.A.6 - Bits are grouped to represent abstractions. These abstractions include, but are not limited to, numbers, characters, and color.	Chapter 10, Lesson 4
DAT-1.A.7 - The same sequence of bits may represent different types of data in different contexts.	Chapter 10, Lesson 4
DAT-1.A.8 - Analog data have values that change smoothly, rather than in discrete intervals, over time. Some examples of analog data include pitch and volume of music, colors of a painting, or position of a sprinter during a race.	Chapter 4, Lesson 1 Chapter 17, Lesson 4
DAT-1.A.9 - The use of digital data to approximate realworld analog data is an example of abstraction.	Chapter 4, Lesson 1 Chapter 17, Lesson 4
DAT-1.A.10 - Analog data can be closely approximated digitally using a sampling technique, which means measuring values of the analog signal at regular intervals called samples. The samples are measured to figure out the exact bits required to store each sample.	Chapter 4, Lesson 1 Chapter 17, Lesson 4
DAT-1.B.1 - In many programming languages, integers are represented by a fixed number of bits, which limits the range of integer values and mathematical operations on those values. This limitation can result in overflow or other errors.	Chapter 4, Lesson 3
DAT-1.B.2 - Other programming languages provide an abstraction through which the size of representable integers is limited only by the size of the computer's memory; this is the case for the language defined in the exam reference sheet.	Chapter 4, Lesson 3
DAT-1.B.3 - In programming languages, the fixed number of bits used to represent real numbers limits the range and mathematical operations on these values; this limitation can result in round-off and other errors. Some real numbers are represented as approximations in computer storage.	Chapter 4, Lesson 3
DAT-1.C.1 - Number bases, including binary and decimal, are used to represent data.	Chapter 10, Lesson 4

DAT-1.C.2 - Binary (base 2) uses only combinations of the digits zero and one.	Chapter 10, Lesson 4
DAT-1.C.3 - Decimal (base 10) uses only combinations of the digits 0 – 9.	Chapter 10, Lesson 4
DAT-1.C.4 - As with decimal, a digit's position in the binary sequence determines its numeric value. The numeric value is equal to the bit's value (0 or 1) multiplied by the place value of its position.	Chapter 10, Lesson 4
DAT-1.C.5 - The place value of each position is determined by the base raised to the power of the position. Positions are numbered starting at the rightmost position with 0 and increasing by 1 for each subsequent position to the left.	Chapter 10, Lesson 4
TOPIC 2.2: Data Compression	
DAT-1.D.1 - Data compression can reduce the size (number of bits) of transmitted or stored data.	Chapter 17, Lesson 4
DAT-1.D.2 - Fewer bits does not necessarily mean less information.	Chapter 17, Lesson 4
DAT-1.D.3 - The amount of size reduction from compression depends on both the amount of redundancy in the original data representation and the compression algorithm applied.	Chapter 17, Lesson 4
DAT-1.D.4 - Lossless data compression algorithms can usually reduce the number of bits stored or transmitted while guaranteeing complete reconstruction of the original data.	Chapter 17, Lesson 4
DAT-1.D.5 - Lossy data compression algorithms can significantly reduce the number of bits stored or transmitted but only allow reconstruction of an approximation of the original data.	Chapter 17, Lesson 4
DAT-1.D.6 - Lossy data compression algorithms can usually reduce the number of bits stored or transmitted more than lossless compression algorithms.	Chapter 17, Lesson 4
DAT-1.D.7 - In situations where quality or ability to reconstruct the original is maximally important, lossless compression algorithms are typically chosen.	Chapter 17, Lesson 4
DAT-1.D.8 - In situations where minimizing data size or transmission time is maximally important, lossy compression algorithms are typically chosen.	Chapter 17, Lesson 4
TOPIC 2.3: Extracting Information from Data	
DAT-2.A.1 - Information is the collection of facts and patterns extracted from data.	Chapter 17, Lesson 1
DAT-2.A.2 - Data provide opportunities for identifying trends, making connections, and addressing problems.	Chapter 17, Lesson 1
DAT-2.A.3 - Digitally processed data may show correlation between variables. A correlation found in data does not necessarily indicate that a causal relationship exists. Additional research is needed to understand the exact nature of the relationship.	Chapter 17, Lesson 1

DAT-2.A.4 - Often, a single source does not contain the data needed to draw a conclusion. It may be necessary to combine data from a variety of sources to formulate a conclusion.	Chapter 17, Lesson 1
DAT-2.B.1 - Metadata are data about data. For example, the piece of data may be an image, while the metadata may include the date of creation or the file size of the image.	Chapter 17, Lesson 1
DAT-2.B.2 - Changes and deletions made to metadata do not change the primary data.	Chapter 17, Lesson 1
DAT-2.B.3 - Metadata are used for finding, organizing, and managing information.	Chapter 17, Lesson 1
DAT-2.B.4 - Metadata can increase the effective use of data or data sets by providing additional information.	Chapter 17, Lesson 1
DAT-2.B.5 - Metadata allow data to be structured and organized.	Chapter 17, Lesson 1
DAT-2.C.1 - The ability to process data depends on the capabilities of the users and their tools.	Chapter 17, Lesson 2
DAT-2.C.2 - Data sets pose challenges regardless of size, such as: * the need to clean data * incomplete data * invalid data * the need to combine data sources	Chapter 17, Lesson 2
DAT-2.C.3 - Depending on how data were collected, they may not be uniform. For example, if users enter data into an open field, the way they choose to abbreviate, spell, or capitalize something may vary from user to user.	Chapter 17, Lesson 2
DAT-2.C.4 - Cleaning data is a process that makes the data uniform without changing their meaning (e.g., replacing all equivalent abbreviations, spellings, and capitalizations with the same word).	Chapter 17, Lesson 2
DAT-2.C.5 - Problems of bias are often created by the type or source of data being collected. Bias is not eliminated by simply collecting more data.	Chapter 17, Lesson 2
DAT-2.C.6 - The size of a data set affects the amount of information that can be extracted from it.	Chapter 17, Lesson 2
DAT-2.C.7 - Large data sets are difficult to process using a single computer and may require parallel systems.	Chapter 17, Lesson 2
DAT-2.C.8 - Scalability of systems is an important consideration when working with data sets, as the computational capacity of a system affects how data sets can be processed and stored.	Chapter 17, Lesson 2
TOPIC 2.4: Using Programs with Data	
DAT-2.D.1 - Programs can be used to process data to acquire information.	Chapter 17, Lesson 3
DAT-2.D.2 - Tables, diagrams, text, and other visual tools can be used to communicate insight and knowledge gained from data.	Chapter 17, Lesson 3
DAT-2.D.3 - Search tools are useful for efficiently finding information.	Chapter 17, Lesson 3

DAT-2.D.4 - Data filtering systems are important tools for finding information and recognizing patterns in data.	Chapter 17, Lesson 3
DAT-2.D.5 - Programs such as spreadsheets help efficiently organize and find trends in information.	Chapter 17, Lesson 3
DAT-2.D.6 - Some processes that can be used to extract or modify information from data include the following: * transforming every element of a data set, such as doubling every element in a list, or adding a parent's email to every student record * filtering a data set, such as keeping only the positive numbers from a list, or keeping only students who signed up for band from a record of all the students * combining or comparing data in some way, such as adding up a list of numbers, or finding the student who has the highest GPA * visualizing a data set through a chart, graph, or other visual representation	Chapter 17, Lesson 3
DAT-2.E.1 - Programs are used in an iterative and interactive way when processing information to allow users to gain insight and knowledge about data.	Chapter 17, Lesson 3
DAT-2.E.2 - Programmers can use programs to filter and clean digital data, thereby gaining insight and knowledge.	Chapter 17, Lesson 3
DAT-2.E.3 - Combining data sources, clustering data, and classifying data are parts of the process of using programs to gain insight and knowledge from data.	Chapter 17, Lesson 3
DAT-2.E.4 - Insight and knowledge can be obtained from translating and transforming digitally represented information.	Chapter 17, Lesson 3
DAT-2.E.5 - Patterns can emerge when data are transformed using programs.	Chapter 17, Lesson 3

BIG IDEA 3: ALGORITHMS AND PROGRAMMING	**CITATION(S)**
TOPIC 3.1: Variables and Assignments	
AAP-1.A.1 - A variable is an abstraction inside a program that can hold a value. Each variable has associated data storage that represents one value at a time, but that value can be a list or other collection that in turn contains multiple values.	Chapter 4, Lesson 1 Chapter 4, Lesson 2
AAP-1.A.2 - Using meaningful variable names helps with the readability of program code and understanding of what values are represented by the variables.	Chapter 4, Lesson 2
AAP-1.A.3 - Some programming languages provide types to represent data, which are referenced using variables. These types include numbers, Booleans, lists, and strings.	Chapter 4, Lesson 2 Chapter 4, Lesson 4
AAP-1.A.4 - Some values are better suited to representation using one type of datum rather than another.	Chapter 4, Lesson 2 Chapter 4, Lesson 4
AAP-1.B.1 - The assignment operator allows a program to change the value represented by a variable.	Chapter 4, Lesson 2

AAP-1.B.2 - The exam reference sheet provides the "<--" operator to use for assignment. (See CED for details)	Chapter 4, Lesson 2 Chapter 21, Lesson 2
AAP-1.B.3 - The value stored in a variable will be the most recent value assigned. (See CED for details)	Chapter 4, Lesson 2
TOPIC 3.2: Data Abstraction	
AAP-1.C.1 - A list is an ordered sequence of elements. For example, [value1, value2, value3, ...] describes a list where value1 is the first element, value2 is the second element, value3 is the third element, and so on.	Chapter 9, Lesson 1
AAP-1.C.2 - An element is an individual value in a list that is assigned a unique index.	Chapter 9, Lesson 1
AAP-1.C.3 - An index is a common method for referencing the elements in a list or string using natural numbers.	Chapter 9, Lesson 1
AAP-1.C.4 - A string is an ordered sequence of characters.	Chapter 4, Lesson 4
AAP-1.D.1 - Data abstraction provides a separation between the abstract properties of a data type and the concrete details of its representation.	Chapter 4, Lesson 1 Chapter 9, Lesson 1
AAP-1.D.2 - Data abstractions manage complexity in programs by giving a collection of data a name without referencing the specific details of the representation.	Chapter 4, Lesson 1 Chapter 9, Lesson 1
AAP-1.D.3 - Data abstractions can be created using lists.	Chapter 4, Lesson 1 Chapter 9, Lesson 1
AAP-1.D.4 - Developing a data abstraction to implement in a program can result in a program that is easier to develop and maintain.	Chapter 4, Lesson 1 Chapter 9, Lesson 1
AAP-1.D.5 - Data abstractions often contain different types of elements.	Chapter 9, Lesson 1
AAP-1.D.6 - The use of lists allows multiple related items to be treated as a single value. Lists are referred to by different names, such as array, depending on the programming language.	Chapter 9, Lesson 1
AAP-1.D.7 - The exam reference sheet provides the notation [value1, value2, value3, ...] to create a list with those values as the first, second, third, and so on items. (See CED for details)	Chapter 9, Lesson 1 Chapter 9, Lesson 2 Chapter 21, Lesson 2
AAP-1.D.8 - The exam reference sheet describes a list structure whose index values are 1 through the number of elements in the list, inclusive. For all list operations, if a list index is less than 1 or greater than the length of the list, an error message is produced and the program will terminate.	Chapter 9, Lesson 1 Chapter 21, Lesson 2
TOPIC 3.3: Mathematical Expressions	
AAP-2.A.1 - An algorithm is a finite set of instructions that accomplish a specific task.	Chapter 15, Lesson 1
AAP-2.A.2 - Beyond visual and textual programming languages, algorithms can be expressed in a variety of ways, such as natural language, diagrams, and pseudocode.	Chapter 15, Lesson 1 Chapter 15, Lesson 2
AAP-2.A.3 - Algorithms executed by programs are implemented using programming languages.	Chapter 15, Lesson 1

AAP-2.A.4 - Every algorithm can be constructed using combinations of sequencing, selection, and iteration.	Chapter 15, Lesson 1
AAP-2.B.1 - Sequencing is the application of each step of an algorithm in the order in which the code statements are given.	Chapter 3, Lesson 3 Chapter 15, Lesson 1
AAP-2.B.2 - A code statement is a part of program code that expresses an action to be carried out.	Chapter 3, Lesson 3
AAP-2.B.3 - An expression can consist of a value, a variable, an operator, or a procedure call that returns a value.	Chapter 4, Lesson 3
AAP-2.B.4 - Expressions are evaluated to produce a single value.	Chapter 4, Lesson 3
AAP-2.B.5 - The evaluation of expressions follows a set order of operations defined by the programming language.	Chapter 4, Lesson 3 Chapter 6, Lesson 4
AAP-2.B.6 - Sequential statements execute in the order they appear in the code segment.	Chapter 3, Lesson 3 Chapter 15, Lesson 1
AAP-2.B.7 - Clarity and readability are important considerations when expressing an algorithm in a programming language.	Chapter 3, Lesson 3 Chapter 4, Lesson 2
AAP-2.C.1 - Arithmetic operators are part of most programming languages and include addition, subtraction, multiplication, division, and modulus operators.	Chapter 4, Lesson 3
AAP-2.C.2 - The exam reference sheet provides a MOD b, which evaluates to the remainder when a is divided by b. Assume that a is an integer greater than or equal to 0 and b is an integer greater than 0. For example, 17 MOD 5 evaluates to 2.	Chapter 4, Lesson 3 Chapter 21, Lesson 2
AAP-2.C.3 - The exam reference sheet provides the arithmetic operators +, -, *, /, and MOD. (See CED for details)	Chapter 4, Lesson 3 Chapter 21, Lesson 2
AAP-2.C.4 - The order of operations used in mathematics applies when evaluating expressions. The MOD operator has the same precedence as the * and / operators.	Chapter 4, Lesson 3
TOPIC 3.4: Strings	
AAP-2.D.1 - String concatenation joins together two or more strings end-to-end to make a new string.	Chapter 4, Lesson 4 Chapter 11, Lesson 1
AAP-2.D.2 - A substring is part of an existing string.	Chapter 11, Lesson 2
TOPIC 3.5: Boolean Expressions	
AAP-2.E.1 - A Boolean value is either true or false.	Chapter 4, Lesson 2 Chapter 6, Lesson 1
AAP-2.E.2 - The exam reference sheet provides the following relational operators: (See CED for details)	Chapter 6, Lesson 1 Chapter 21, Lesson 2
AAP-2.F.1 - The exam reference sheet provides the logical operators NOT, AND, and OR, which evaluate to a Boolean value.	Chapter 6, Lesson 3 Chapter 21, Lesson 2
AAP-2.F.2- The exam reference sheet provides NOT condition (See CED for details)	Chapter 6, Lesson 3 Chapter 21, Lesson 2

AAP-2.F.3 - The exam reference sheet provides condition1 AND condition2 (See CED for details)	Chapter 6, Lesson 3 Chapter 21, Lesson 2
AAP-2.F.4 - The exam reference sheet provides condition1 OR condition2 (See CED for details)	Chapter 6, Lesson 3 Chapter 21, Lesson 2
AAP-2.F.5 - The operand for a logical operator is either a Boolean expression or a single Boolean value.	Chapter 6, Lesson 3
TOPIC 3.6: Conditionals	
AAP-2.G.1 - Selection determines which parts of an algorithm are executed based on a condition being true or false.	Chapter 6, Lesson 2 Chapter 15, Lesson 1
AAP-2.H.1 - Conditional statements, or "if-statements," affect the sequential flow of control by executing different statements based on the value of a Boolean expression.	Chapter 6, Lesson 2
AAP-2.H.2 - The exam reference sheet provides IF(condition) (See CED for details)	Chapter 6, Lesson 2 Chapter 21, Lesson 2
AAP-2.H.3 - The exam reference sheet provides IF(condition) / ELSE (See CED for details)	Chapter 6, Lesson 2 Chapter 21, Lesson 2
TOPIC 3.7: Nested Conditionals	
AAP-2.I.1 - Nested conditional statements consist of conditional statements within conditional statements.	Chapter 6, Lesson 2 Chapter 6, Lesson 3
TOPIC 3.8: Iteration	
AAP-2.J.1 - Iteration is a repeating portion of an algorithm. Iteration repeats a specified number of times or until a given condition is met.	Chapter 8, Lesson 1 Chapter 8, Lesson 2
AAP-2.K.1 - Iteration statements change the sequential flow of control by repeating a set of statements zero or more times, until a stopping condition is met.	Chapter 8, Lesson 1 Chapter 8, Lesson 2
AAP-2.K.2 - The exam reference sheet provides REPEAT n TIMES (See CED for details)	Chapter 8, Lesson 1 Chapter 21, Lesson 2
AAP-2.K.3 - The exam reference sheet provides REPEAT UNTIL(condition) (See CED for details)	Chapter 8, Lesson 2 Chapter 21, Lesson 2
AAP-2.K.4 - In REPEAT UNTIL(condition) iteration, an infinite loop occurs when the ending condition will never evaluate to true.	Chapter 8, Lesson 2
AAP-2.K.5 - In REPEAT UNTIL(condition) iteration, if the conditional evaluates to true initially, the loop body is not executed at all, due to the condition being checked before the loop.	Chapter 8, Lesson 2
TOPIC 3.9: Developing Algorithms	
AAP-2.L.1 - Algorithms can be written in different ways and still accomplish the same tasks.	Chapter 15, Lesson 3
AAP-2.L.2 - Algorithms that appear similar can yield different side effects or results.	Chapter 15, Lesson 3 Chapter 16, Lesson 2
AAP-2.L.3 - Some conditional statements can be written as equivalent Boolean expressions.	Chapter 6, Lesson 3

AAP-2.L.4 - Some Boolean expressions can be written as equivalent conditional statements.	Chapter 6, Lesson 3
AAP-2.L.5 - Different algorithms can be developed or used to solve the same problem.	Chapter 15, Lesson 3
AAP-2.M.1 - Algorithms can be created from an idea, by combining existing algorithms, or by modifying existing algorithms.	Chapter 15, Lesson 3
AAP-2.M.2 - Knowledge of existing algorithms can help in constructing new ones. Some existing algorithms include: * determining the maximum or minimum value of two or more numbers * computing the sum or average of two or more numbers * identifying if an integer is or is not evenly divisible by another integer * determining a robot's path through a maze	Chapter 15, Lesson 3 Chapter 21, Lesson 3
AAP-2.M.3 - Using existing correct algorithms as building blocks for constructing another algorithm has benefits such as reducing development time, reducing testing, and simplifying the identification of errors.	Chapter 10, Lesson 1
TOPIC 3.10: Lists	
AAP-2.N.1 - The exam reference sheet provides basic operations on lists (See CED for details)	Chapter 9, Lesson 1 Chapter 9, Lesson 2 Chapter 21, Lesson 2
AAP-2.N.2 - List procedures are implemented in accordance with the syntax rules of the programming language.	Chapter 9, Lesson 2
AAP-2.O.1 - Traversing a list can be a complete traversal, where all elements in the list are accessed, or a partial traversal, where only a portion of elements are accessed.	Chapter 9, Lesson 3
AAP-2.O.2 - Iteration statements can be used to traverse a list.	Chapter 9, Lesson 3
AAP-2.O.3 - The exam reference sheet provides FOR EACH item IN aList (See CED for details)	Chapter 9, Lesson 3 Chapter 21, Lesson 2
AAP-2.O.4 - Knowledge of existing algorithms that use iteration can help in constructing new algorithms. Some examples of existing algorithms that are often used with lists include: * determining a minimum or maximum value in a list * computing a sum or average of a list of numbers	Chapter 9, Lesson 3
AAP-2.O.5 - Linear search or sequential search algorithms check each element of a list, in order, until the desired value is found or all elements in the list have been checked.	Chapter 9, Lesson 4
TOPIC 3.11: Binary Search	
AAP-2.P.1 - The binary search algorithm starts at the middle of a sorted data set of numbers and eliminates half of the data; this process repeat until the desired value is found or all elements have been eliminated.	Chapter 9, Lesson 4
AAP-2.P.2 - Data must be in sorted order to use the binary search algorithm.	Chapter 9, Lesson 4

AAP-2.P.3 - Binary search is often more efficient than sequential/linear search when applied to sorted data.	Chapter 9, Lesson 4
TOPIC 3.12: Calling Procedures	
AAP-3.A.1 - A procedure is a named group of programming instructions that may have parameters and return values.	Chapter 12, Lesson 1 Chapter 12, Lesson 2
AAP-3.A.2 - Procedures are referred to by different names, such as method or function, depending on the programming language.	Chapter 12, Lesson 1
AAP-3.A.3 - Parameters are input variables of a procedure. Arguments specify the values of the parameters when a procedure is called.	Chapter 12, Lesson 2
AAP-3.A.4 - A procedure call interrupts the sequential execution of statements, causing the program to execute the statements within the procedure before continuing. Once the last statement in the procedure (or a return statement) has executed, flow of control is returned to the point immediately following where the procedure was called.	Chapter 12, Lesson 1
AAP-3.A.5 - The exam reference sheet provides procName(arg1, arg2, ...) as a way to call (See CED for details)	Chapter 12, Lesson 2 Chapter 21, Lesson 2
AAP-3.A.6 - The exam reference sheet provides the procedure DISPLAY(expression) (See CED for details)	Chapter 5, Lesson 1 Chapter 21, Lesson 2
AAP-3.A.7 - The exam reference sheet provides RETURN(expression) (See CED for details)	Chapter 12, Lesson 2 Chapter 21, Lesson 2
AAP-3.A.8 - The exam reference sheet provides result procName(arg1, arg2, ...) to assign to result the "value of the procedure" being returned (See CED for details)	Chapter 12, Lesson 2 Chapter 21, Lesson 2
AAP-3.A.9 - The exam reference sheet provides procedure INPUT() (See CED for details)	Chapter 5, Lesson 2 Chapter 21, Lesson 2
TOPIC 3.13: Developing Procedures	
AAP-3.B.1 - One common type of abstraction is procedural abstraction, which provides a name for a process and allows a procedure to be used only knowing what it does, not how it does it.	Chapter 12, Lesson 4
AAP-3.B.2 - Procedural abstraction allows a solution to a large problem to be based on the solutions of smaller subproblems. This is accomplished by creating procedures to solve each of the subproblems.	Chapter 12, Lesson 4
AAP-3.B.3 - The subdivision of a computer program into separate subprograms is called modularity.	Chapter 12, Lesson 4
AAP-3.B.4 - A procedural abstraction may extract shared features to generalize functionality instead of duplicating code. This allows for program code reuse, which helps manage complexity.	Chapter 12, Lesson 4
AAP-3.B.5 - Using parameters allows procedures to be generalized, enabling the procedures to be reused with a range of input values or arguments.	Chapter 12, Lesson 2 Chapter 12, Lesson 4
AAP-3.B.6 - Using procedural abstraction helps improve code readability.	Chapter 12, Lesson 4

AAP-3.B.7 - Using procedural abstraction in a program allows programmers to change the internals of the procedure (to make it faster, more efficient, use less storage, etc.) without needing to notify users of the change as long as what the procedure does is preserved.	Chapter 12, Lesson 4
AAP-3.C.1 - The exam reference sheet provides PROCEDURE procName(parameter1, parameter2, ...) (See CED for details)	Chapter 12, Lesson 2 Chapter 21, Lesson 2
AAP-3.C.2 - The exam reference sheet provides PROCEDURE procName(parameter1, parameter2, ...) with RETURN (See CED for details)	Chapter 12, Lesson 2 Chapter 21, Lesson 2
TOPIC 3.14: Libraries	
AAP-3.D.1 - A software library contains procedures that may be used in creating new programs.	Chapter 10, Lesson 1
AAP-3.D.2 - Existing code segments can come from internal or external sources, such as libraries or previously written code.	Chapter 10, Lesson 1
AAP-3.D.3 - The use of libraries simplifies the task of creating complex programs.	Chapter 10, Lesson 1
AAP-3.D.4 - Application program interfaces (APIs) are specifications for how the procedures in a library behave and can be used.	Chapter 10, Lesson 1
AAP-3.D.5 - Documentation for an API/library is necessary in understanding the behaviors provided by the API/library and how to use them.	Chapter 10, Lesson 1
TOPIC 3.15: Random Values	
AAP-3.E.1 - The exam reference sheet provides RANDOM(a, b) (See CED for details)	Chapter 10, Lesson 2 Chapter 21, Lesson 2
AAP-3.E.2 - Using random number generation in a program means each execution may produce a different result.	Chapter 10, Lesson 2
TOPIC 3.16: Simulations	
AAP-3.F.1 - Simulations are abstractions of more complex objects or phenomena for a specific purpose.	Chapter 16, Lesson 3
AAP-3.F.2 - A simulation is a representation that uses varying sets of values to reflect the changing state of a phenomenon.	Chapter 16, Lesson 3
AAP-3.F.3 - Simulations often mimic real-world events with the purpose of drawing inferences, allowing investigation of a phenomenon without the constraints of the real world.	Chapter 16, Lesson 3
AAP-3.F.4 - The process of developing an abstract simulation involves removing specific details or simplifying functionality.	Chapter 16, Lesson 3
AAP-3.F.5 - Simulations can contain bias derived from the choices of real-world elements that were included or excluded.	Chapter 16, Lesson 3
AAP-3.F.6 - Simulations are most useful when real-world events are impractical for experiments (e.g., too big, too small, too fast, too slow, too expensive, or too dangerous).	Chapter 16, Lesson 3
AAP-3.F.7 - Simulations facilitate the formulation and refinement of hypotheses related to the objects or phenomena under consideration.	Chapter 16, Lesson 3

AAP-3.F.8 - Random number generators can be used to simulate the variability that exists in the real world.	Chapter 16, Lesson 3
TOPIC 3.17: Algorithmic Efficiency	
AAP-4.A.1 - A problem is a general description of a task that can (or cannot) be solved algorithmically. An instance of a problem also includes specific input. For example, sorting is a problem; sorting the list (2,3,1,7) is an instance of the problem.	Chapter 16, Lesson 2
AAP-4.A.2 - A decision problem is a problem with a yes/no answer (e.g., is there a path from A to B?). An optimization problem is a problem with the goal of finding the "best" solution among many (e.g., what is the shortest path from A to B?).	Chapter 16, Lesson 2
AAP-4.A.3 - Efficiency is an estimation of the amount of computational resources used by an algorithm. Efficiency is typically expressed as a function of the size of the input.	Chapter 16, Lesson 1
AAP-4.A.4 - An algorithm’s efficiency is determined through formal or mathematical reasoning.	Chapter 16, Lesson 1
AAP-4.A.5 - An algorithm’s efficiency can be informally measured by determining the number of times a statement or group of statements executes.	Chapter 16, Lesson 1
AAP-4.A.6 - Different correct algorithms for the same problem can have different efficiencies.	Chapter 16, Lesson 1
AAP-4.A.7 - Algorithms with a polynomial efficiency or slower (constant, linear, square, cube, etc.) are said to run in a reasonable amount of time. Algorithms with exponential or factorial efficiencies are examples of algorithms that run in an unreasonable amount of time.	Chapter 16, Lesson 1
AAP-4.A.8 - Some problems cannot be solved in a reasonable amount of time because there is no efficient algorithm for solving them. In these cases, approximate solutions are sought.	Chapter 16, Lesson 2
AAP-4.A.9 - A heuristic is an approach to a problem that produces a solution that is not guaranteed to be optimal but may be used when techniques that are guaranteed to always find an optimal solution are impractical.	Chapter 16, Lesson 2
TOPIC 3.18: Undecidable Problems	
AAP-4.B.1 - A decidable problem is a decision problem for which an algorithm can be written to produce a correct output for all inputs (e.g., "Is the number even?").	Chapter 16, Lesson 2
AAP-4.B.2 - An undecidable problem is one for which no algorithm can be constructed that is always capable of providing a correct yes-or-no answer.	Chapter 16, Lesson 2
AAP-4.B.3 - An undecidable problem may have some instances that have an algorithmic solution, but there is no algorithmic solution that could solve all instances of the problem.	Chapter 16, Lesson 2

BIG IDEA 4: COMPUTER SYSTEMS AND NETWORKS	CITATION(S)
TOPIC 1.1: The Internet	
CSN-1.A.1 - A computing device is a physical artifact that can run a program. Some examples include computers, tablets, servers, routers, and smart sensors.	Chapter 1, Lesson 2
CSN-1.A.2 - A computing system is a group of computing devices and programs working together for a common purpose.	Chapter 2, Lesson 1
CSN-1.A.3 - A computer network is a group of interconnected computing devices capable of sending or receiving data.	Chapter 2, Lesson 1
CSN-1.A.4 - A computer network is a type of computing system.	Chapter 2, Lesson 1
CSN-1.A.5 - A path between two computing devices on a computer network (a sender and a receiver) is a sequence of directly connected computing devices that begins at the sender and ends at the receiver.	Chapter 2, Lesson 2 Chapter 2, Lesson 3
CSN-1.A.6 - Routing is the process of finding a path from sender to receiver.	Chapter 2, Lesson 2 Chapter 2, Lesson 3
CSN-1.A.7 - The bandwidth of a computer network is the maximum amount of data that can be sent in a fixed amount of time.	Chapter 2, Lesson 1
CSN-1.A.8 - Bandwidth is usually measured in bits per second.	Chapter 2, Lesson 1
CSN-1.B.1 - The Internet is a computer network consisting of interconnected networks that use standardized, open (nonproprietary) communication protocols.	Chapter 2, Lesson 2
CSN-1.B.2 - Access to the Internet depends on the ability to connect a computing device to an Internet connected device.	Chapter 2, Lesson 2 Chapter 2, Lesson 3
CSN-1.B.3 - A protocol is an agreed-upon set of rules that specify the behavior of a system.	Chapter 2, Lesson 2
CSN-1.B.4 - The protocols used in the Internet are open, which allows users to easily connect additional computing devices to the Internet.	Chapter 2, Lesson 2
CSN-1.B.5 - Routing on the Internet is usually dynamic; it is not specified in advance.	Chapter 2, Lesson 2 Chapter 2, Lesson 3
CSN-1.B.6 - The scalability of a system is the capacity for the system to change in size and scale to meet new demands.	Chapter 2, Lesson 3
CSN-1.B.7 - The Internet was designed to be scalable.	Chapter 2, Lesson 3
CSN-1.C.1 - Information is passed through the Internet as a data stream. Data streams contain chunks of data, which are encapsulated in packets.	Chapter 2, Lesson 2
CSN-1.C.2 - Packets contain a chunk of data and metadata used for routing the packet between the origin and the destination on the Internet, as well as for data reassembly.	Chapter 2, Lesson 2
CSN-1.C.3 - Packets may arrive at the destination in order, out of order, or not at all.	Chapter 2, Lesson 2

CSN-1.C.4 - IP, TCP, and UDP are common protocols used on the Internet.	Chapter 2, Lesson 2
CSN-1.D.1 - The World Wide Web is a system of linked pages, programs, and files.	Chapter 2, Lesson 2
CSN-1.D.2 - HTTP is a protocol used by the World Wide Web.	Chapter 2, Lesson 2
CSN-1.D.3 - The World Wide Web uses the Internet.	Chapter 2, Lesson 2
TOPIC 4.2: Fault Tolerance	
CSN-1.E.1 - The Internet has been engineered to be fault tolerant, with abstractions for routing and transmitting data.	Chapter 2, Lesson 3
CSN-1.E.2 - Redundancy is the inclusion of extra components that can be used to mitigate failure of a system if other components fail.	Chapter 2, Lesson 3
CSN-1.E.3 - One way to accomplish network redundancy is by having more than one path between any two connected devices.	Chapter 2, Lesson 3
CSN-1.E.4 - If a particular device or connection on the Internet fails, subsequent data will be sent via a different route, if possible.	Chapter 2, Lesson 3
CSN-1.E.5 - When a system can support failures and still continue to function, it is called fault-tolerant. This is important because elements of complex systems fail at unexpected times, often in groups, and fault tolerance allows users to continue to use the network.	Chapter 2, Lesson 3
CSN-1.E.6 - Redundancy within a system often requires additional resources but can provide the benefit of fault tolerance.	Chapter 2, Lesson 3
CSN-1.E.7 - The redundancy of routing options between two points increases the reliability of the Internet and helps it scale to more devices and more people.	Chapter 2, Lesson 3
TOPIC 4.3: Parallel and Distributed Computing	
CSN-2.A.1 - Sequential computing is a computational model in which operations are performed in order one at a time.	Chapter 2, Lesson 4
CSN-2.A.2 - Parallel computing is a computational model where the program is broken into multiple smaller sequential computing operations, some of which are performed simultaneously.	Chapter 2, Lesson 4
CSN-2.A.3 - Distributed computing is a computational model in which multiple devices are used to run a program.	Chapter 2, Lesson 4
CSN-2.A.4 - Comparing efficiency of solutions can be done by comparing the time it takes them to perform the same task.	Chapter 2, Lesson 4
CSN-2.A.5 - A sequential solution takes as long as the sum of all of its steps.	Chapter 2, Lesson 4
CSN-2.A.6 - A parallel computing solution takes as long as its sequential tasks plus the longest of its parallel tasks.	Chapter 2, Lesson 4
CSN-2.A.7 - The "speedup" of a parallel solution is measured in the time it took to complete the task sequentially divided by the time it took to complete the task when done in parallel.	Chapter 2, Lesson 4

CSN-2.B.1 - Parallel computing consists of a parallel portion and a sequential portion.	Chapter 2, Lesson 4
CSN-2.B.2 - Solutions that use parallel computing can scale more effectively than solutions that use sequential computing.	Chapter 2, Lesson 4
CSN-2.B.3 - Distributed computing allows problems to be solved that could not be solved on a single computer because of either the processing time or storage needs involved.	Chapter 2, Lesson 4
CSN-2.B.4 - Distributed computing allows much larger problems to be solved quicker than they could be solved using a single computer.	Chapter 2, Lesson 4
CSN-2.B.5 - When increasing the use of parallel computing in a solution, the efficiency of the solution is still limited by the sequential portion. This means that at some point, adding parallel portions will no longer meaningfully increase efficiency.	Chapter 2, Lesson 4

BIG IDEA 5: IMPACT OF COMPUTING	**CITATION(S)**
TOPIC 5.1: Beneficial and Harmful Effects	
IOC-1.A.1 - People create computing innovations.	Chapter 18, Lesson 2
IOC-1.A.2 - The way people complete tasks often changes to incorporate new computing innovations.	Chapter 18, Lesson 2
IOC-1.A.3 - Not every effect of a computing innovation is anticipated in advance.	Chapter 18, Lesson 2
IOC-1.A.4 - A single effect can be viewed as both beneficial and harmful by different people, or even by the same person.	Chapter 18, Lesson 2
IOC-1.A.5 - Advances in computing have generated and increased creativity in other fields, such as medicine, engineering, communications, and the arts.	Chapter 18, Lesson 2
IOC-1.B.1 - Computing innovations can be used in ways that their creators had not originally intended: * The World Wide Web was originally intended only for rapid and easy exchange of information within the scientific community. * Targeted advertising is used to help businesses, but it can be misused at both individual and aggregate levels. * Machine learning and data mining have enabled innovation in medicine, business, and science, but information discovered in this way has also been used to discriminate against groups of individuals.	Chapter 18, Lesson 2
IOC-1.B.2 - Some of the ways computing innovations can be used may have a harmful impact on society, the economy, or culture.	Chapter 18, Lesson 2
IOC-1.B.3 - Responsible programmers try to consider the unintended ways their computing innovations can be used and the potential beneficial and harmful effects of these new uses.	Chapter 18, Lesson 2
IOC-1.B.4 - It is not possible for a programmer to consider all the ways a computing innovation can be used.	Chapter 18, Lesson 2

IOC-1.B.5 - Computing innovations have often had unintended beneficial effects by leading to advances in other fields.	Chapter 18, Lesson 2
IOC-1.B.6 - Rapid sharing of a program or running a program with a large number of users can result in significant impacts beyond the intended purpose or control of the programmer.	Chapter 18, Lesson 2
TOPIC 5.2: Digital Divide	
IOC-1.C.1 - Internet access varies between socioeconomic, geographic, and demographic characteristics, as well as between countries.	Chapter 18, Lesson 1
IOC-1.C.2 - The "digital divide" refers to differing access to computing devices and the Internet, based on socioeconomic, geographic, or demographic characteristics.	Chapter 18, Lesson 1
IOC-1.C.3 - The digital divide can affect both groups and individuals.	Chapter 18, Lesson 1
IOC-1.C.4 - The digital divide raises issues of equity, access, and influence, both globally and locally.	Chapter 18, Lesson 1
IOC-1.C.5 - The digital divide is affected by the actions of individuals, organizations, and governments.	Chapter 18, Lesson 1
TOPIC 5.3: Computing Bias	
IOC-1.D.1 - Computing innovations can reflect existing human biases because of biases written into the algorithms or biases in the data used by the innovation.	Chapter 18, Lesson 1
IOC-1.D.2 - Programmers should take action to reduce bias in algorithms used for computing innovations as a way of combating existing human biases.	Chapter 18, Lesson 1
IOC-1.D.3 - Biases can be embedded at all levels of software development.	Chapter 18, Lesson 1
TOPIC 5.4: Crowdsourcing	
IOC-1.E.1 - Widespread access to information and public data facilitates the identification of problems, development of solutions, and dissemination of results.	Chapter 18, Lesson 3
IOC-1.E.2 - Science has been affected by using distributed and "citizen science" to solve scientific problems.	Chapter 18, Lesson 3
IOC-1.E.3 - Citizen science is scientific research conducted in whole or part by distributed individuals, many of whom may not be scientists, who contribute relevant data to research using their own computing devices.	Chapter 18, Lesson 3
IOC-1.E.4 - Crowdsourcing is the practice of obtaining input or information from a large number of people via the Internet.	Chapter 18, Lesson 3
IOC-1.E.5 - Human capabilities can be enhanced by collaboration via computing.	Chapter 18, Lesson 3
IOC-1.E.6 - Crowdsourcing offers new models for collaboration, such as connecting businesses or social causes with funding.	Chapter 18, Lesson 3

TOPIC 5.5: Legal and Ethical Concerns	
IOC-1.F.1 - Material created on a computer is the intellectual property of the creator or an organization.	Chapter 19, Lesson 2 Chapter 19, Lesson 3
IOC-1.F.2 - Ease of access and distribution of digitized information raises intellectual property concerns regarding ownership, value, and use.	Chapter 19, Lesson 3
IOC-1.F.3 - Measures should be taken to safeguard intellectual property.	Chapter 19, Lesson 2 Chapter 19, Lesson 3
IOC-1.F.4 - The use of material created by someone else without permission and presented as one's own is plagiarism and may have legal consequences.	Chapter 19, Lesson 2
IOC-1.F.5 - Some examples of legal ways to use materials created by someone else include: * Creative Commons—a public copyright license that enables the free distribution of an otherwise copyrighted work. This is used when the content creator wants to give others the right to share, use, and build upon the work they have created. * open source—programs that are made freely available and may be redistributed and modified * open access—online research output free of any and all restrictions on access and free of many restrictions on use, such as copyright or license restrictions	Chapter 19, Lesson 2 Chapter 19, Lesson 3
IOC-1.F.6 - The use of material created by someone other than you should always be cited.	Chapter 19, Lesson 2 Chapter 19, Lesson 3
IOC-1.F.7 - Creative Commons, open source, and open access have enabled broad access to digital information.	Chapter 19, Lesson 2 Chapter 19, Lesson 3
IOC-1.F.8 - As with any technology or medium, using computing to harm individuals or groups of people raises legal and ethical concerns.	Chapter 19, Lesson 1
IOC-1.F.9 - Computing can play a role in social and political issues, which in turn often raises legal and ethical concerns.	Chapter 19, Lesson 1
IOC-1.F.10 - The digital divide raises ethical concerns around computing.	Chapter 18, Lesson 1 Chapter 19, Lesson 1
IOC-1.F.11 - Computing innovations can raise legal and ethical concerns. Some examples of these include: * the development of software that allows access to digital media downloads and streaming * the development of algorithms that include bias * the existence of computing devices that collect and analyze data by continuously monitoring activities	Chapter 18, Lesson 1 Chapter 18, Lesson 2 Chapter 19, Lesson 1

Topic 5.6: Safe Computing	
IOC-2.A.1 - Personally identifiable information (PII) is information about an individual that identifies, links, relates, or describes them. Examples of PII include: * Social Security number * age * race * phone number(s) * medical information * financial information	Chapter 20, Lesson 1
IOC-2.A.2 - Search engines can record and maintain a history of searches made by users.	Chapter 20, Lesson 1
IOC-2.A.3 - Websites can record and maintain a history of individuals who have viewed their pages.	Chapter 20, Lesson 1
IOC-2.A.4 - Devices, websites, and networks can collect information about a user's location.	Chapter 20, Lesson 1
IOC-2.A.5 - Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions.	Chapter 20, Lesson 1
IOC-2.A.6 - Search engines can use search history to suggest websites or for targeted marketing.	Chapter 20, Lesson 1
IOC-2.A.7 - Disparate personal data, such as geolocation, cookies, and browsing history, can be aggregated to create knowledge about an individual.	Chapter 20, Lesson 1
IOC-2.A.8 - PII and other information placed online can be used to enhance a user's online experiences.	Chapter 20, Lesson 1
IOC-2.A.9 - PII stored online can be used to simplify making online purchases.	Chapter 20, Lesson 1
IOC-2.A.10 - Commercial and governmental curation of information may be exploited if privacy and other protections are ignored.	Chapter 20, Lesson 1
IOC-2.A.11 - Information placed online can be used in ways that were not intended and that may have a harmful impact. For example, an email message may be forwarded, tweets can be retweeted, and social media posts can be viewed by potential employers.	Chapter 20, Lesson 1
IOC-2.A.12 - PII can be used to stalk or steal the identity of a person or to aid in the planning of other criminal acts.	Chapter 20, Lesson 1
IOC-2.A.13 - Once information is placed online, it is difficult to delete.	Chapter 20, Lesson 1
IOC-2.A.14 - Programs can collect your location and record where you have been, how you got there, and how long you were at a given location.	Chapter 20, Lesson 1
IOC-2.A.15 - Information posted to social media services can be used by others. Combining information posted on social media and other sources can be used to deduce private information about you.	Chapter 20, Lesson 1

IOC-2.B.1 - Authentication measures protect devices and information from unauthorized access. Examples of authentication measures include strong passwords and multifactor authentication.	Chapter 20, Lesson 2
IOC-2.B.2 - A strong password is something that is easy for a user to remember but would be difficult for someone else to guess based on knowledge of that user.	Chapter 20, Lesson 2
IOC-2.B.3 - Multifactor authentication is a method of computer access control in which a user is only granted access after successfully presenting several separate pieces of evidence to an authentication mechanism, typically in at least two of the following categories: knowledge (something they know), possession (something they have), and inherence (something they are).	Chapter 20, Lesson 2
IOC-2.B.4 - Multifactor authentication requires at least two steps to unlock protected information; each step adds a new layer of security that must be broken to gain unauthorized access.	Chapter 20, Lesson 2
IOC-2.B.5 - Encryption is the process of encoding data to prevent unauthorized access. Decryption is the process of decoding the data. Two common encryption approaches are: * Symmetric key encryption involves one key for both encryption and decryption. * Public key encryption pairs a public key for encryption and a private key for decryption. The sender does not need the receiver's private key to encrypt a message, but the receiver's private key is required to decrypt the message.	Chapter 20, Lesson 2
IOC-2.B.6 - Certificate authorities issue digital certificates that validate the ownership of encryption keys used in secure communications and are based on a trust model.	Chapter 20, Lesson 2
IOC-2.B.7 - Computer virus and malware scanning software can help protect a computing system against infection.	Chapter 20, Lesson 2
IOC-2.B.8 - A computer virus is a malicious program that can copy itself and gain access to a computer in an unauthorized way. Computer viruses often attach themselves to legitimate programs and start running independently on a computer.	Chapter 20, Lesson 2
IOC-2.B.9 - Malware is software intended to damage a computing system or to take partial control over its operation.	Chapter 20, Lesson 2
IOC-2.B.10 - All real-world systems have errors or design flaws that can be exploited to compromise them. Regular software updates help fix errors that could compromise a computing system.	Chapter 20, Lesson 2
IOC-2.B.11 - Users can control the permissions programs have for collecting user information. Users should review the permission settings of programs to protect their privacy.	Chapter 20, Lesson 1
IOC-2.C.1 - Phishing is a technique that attempts to trick a user into providing personal information. That personal information can then be used to access sensitive online resources, such as bank accounts and emails.	Chapter 20, Lesson 3

IOC-2.C.2 - Keylogging is the use of a program to record every keystroke made by a computer user in order to gain fraudulent access to passwords and other confidential information.	Chapter 20, Lesson 3
IOC-2.C.3 - Data sent over public networks can be intercepted, analyzed, and modified. One way that this can happen is through a rogue access point.	Chapter 20, Lesson 3
IOC-2.C.4 - A rogue access point is a wireless access point that gives unauthorized access to secure networks.	Chapter 20, Lesson 3
IOC-2.C.5 - A malicious link can be disguised on a web page or in an email message.	Chapter 20, Lesson 3
IOC-2.C.6 - Unsolicited emails, attachments, links, and forms in emails can be used to compromise the security of a computing system. These can come from unknown senders or from known senders whose security has been compromised.	Chapter 20, Lesson 3
IOC-2.C.7 - Untrustworthy (often free) downloads from freeware or shareware sites can contain malware.	Chapter 20, Lesson 3