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 2020
Standards Link:	AP CSP Exam Description Page

CompuScholar Course Details:

Course Title:	Computer Science Foundations
Course ISBN:	978-1-946113-02-3
Course Year:	2022

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
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 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 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 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 3
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
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
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
CRD-2.J.3 - Program requirements are needed to identify appropriate defined inputs for testing.	Chapter 7, Lesson 2

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 Chapter 10, Lesson 4
DAT-1.A.2 - Computing devices represent data digitally, meaning that the lowest-level components of any value are bits.	Chapter 4, Lesson 1 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
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 2, Lesson 3 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 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 4, Lesson 2
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 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 Chapter 15, Lesson 3
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 3
AAP-2.A.3 - Algorithms executed by programs are implemented using programming languages.	Chapter 15, Lesson 1 Chapter 15, Lesson 3
AAP-2.A.4 - Every algorithm can be constructed using combinations of sequencing, selection, and iteration.	Chapter 15, Lesson 1 Chapter 15, Lesson 3

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
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 6, Lesson 4
AAP-2.B.6 - Sequential statements execute in the order they appear in the code segment.	Chapter 3, Lesson 3
AAP-2.B.7 - Clarity and readability are important considerations when expressing an algorithm in a programming language.	Chapter 3, Lesson 3
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
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 6, Lesson 4
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 3.5: 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
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
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 Chapter 21, 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 Chapter 21, 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 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
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 1
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 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 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 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 3
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 3
IOC-1.F.6 - The use of material created by someone other than you should always be cited.	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 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 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 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 2
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