



The College Board

AP[®] Computer Science Principles Draft Curriculum Framework

June 2014

This document is based upon work supported by the National Science Foundation, grant CNS-0938336. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

© 2014 The College Board. All rights reserved. Computer Science Principles is a pilot course under development. It is not an official Advanced Placement course currently offered by the College Board. **This is a draft publication and is not intended for distribution.**

Overview of the Curriculum Framework

The AP Computer Science Principles course is designed to be equivalent to a first-semester introductory college computing course. This curriculum framework provides a detailed description of the course content. The key sections of this framework are described below.

- The **computational thinking practices** capture important aspects of the work that computer scientists engage in, at the level of competence expected of AP Computer Science Principles students. The practices help students coordinate and make sense of knowledge to accomplish a goal or task. They enable students to engage with the course content by developing computational artifacts and analyzing data, information, or knowledge represented for computational use. In addition, the practices and skills require students to learn to collaborate to build computational artifacts and communicate their purpose. Because the AP Computer Science Principles content and the computational practices and skills are equally important, each learning objective directly correlates to a computational thinking practice or skill. This correlation to a computational thinking practice or skill is denoted at the end of a learning objective. For example, [P1] represents a correlation to Computational Thinking Practice 1, which is Connecting Computing.
- The key concepts and related content that define the course and exam are organized around seven **big ideas**, which encompass fundamental ideas foundational to computing. These big ideas connect students to a curriculum scope that includes the art of programming but is not programming-centric. A set of **essential questions** are included under each big idea. These questions are large in scope and are provided to help students consider connections to the content of the big ideas. They highlight what is needed for learning the core content in each big idea. Additionally, each of the big ideas contain **enduring understandings**, which specify core concepts that students should retain from their learning experiences.
- Each enduring understanding is aligned with at least one or more **learning objectives** that provide a detailed articulation of what students are expected to be able to do by the end of the course. The learning objectives integrate a computational thinking practice or skill with specific content, and provide clear information about how students will be expected to demonstrate their knowledge and abilities. They are numbered to correspond with the big ideas and enduring understandings (e.g., LO 1.1.1). **The learning objectives will be the target of assessment on the AP Computer Science Principles performance tasks and AP Exam.**
- Next to each learning objective is a listing of **essential knowledge statements**. These statements specify facts or content that students must know in order to be able to successfully demonstrate understanding of the learning objectives. These additional underlying content components are listed numerically in the column next to the supported learning objective, and each one includes one or more bulleted statements describing further content details. All examples and content references are considered to be required and may be the focus of exam questions. For example, the following essential knowledge statements correspond to Learning Objective 1.1.1, *Apply a creative development process when creating computational artifacts*. [P2]:

1.1.1A A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.

1.1.1B Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.

Exclusion statements are included in various locations of the framework. These statements provide further clarity about the scope of a particular learning objective or essential knowledge statement. They specify content that will not be assessed on the exam because it is outside the scope of the course. For example:

- **Exclusion Statement (LO 4.2.1):** Any discussion of nondeterministic polynomial (NP) is beyond the scope of this course and the AP Exam.

This statement applies to learning objective (LO) 4.2.1 and it specifies content that is outside the scope of the course and exam.

Computational Thinking Practices

P1: Connecting Computing

Developments in computing have far-reaching effects on society and have led to significant innovations. The developments have implications for individuals, society, commercial markets, and innovation. Students in this course study these effects, and they learn to draw connections between different computing concepts. Students are expected to:

- Identify impacts of computing;
- Describe connections between people and computing; and
- Explain connections between computing concepts.

P2: Creating Computational Artifacts

Computing is a creative discipline in which creation takes many forms, such as remixing digital music, generating animations, developing websites, and writing programs. Students in this course engage in the creative aspects of computing by designing and developing interesting computational artifacts, as well as by applying computing techniques to creatively solve problems. Students are expected to:

- Create an artifact with a practical, personal, or societal intent;
- Select appropriate techniques to develop a computational artifact; and
- Use appropriate algorithmic and information-management principles.

P3: Abstracting

Computational thinking requires understanding and applying abstraction at multiple levels, such as privacy in social networking applications, logic gates and bits, and the human genome project. Students in this course use abstraction to develop models and simulations of natural and artificial phenomena, use them to make predictions about the world, and analyze their efficacy and validity. Students are expected to:

- Explain how data, information, or knowledge is represented for computational use;
- Explain how abstractions are used in computation or modeling;

- Identify abstractions; and
- Describe modeling in a computational context.

P4: Analyzing Problems and Artifacts

The results and artifacts of computation and the computational techniques and strategies that generate them can be understood both intrinsically for what they are as well as for what they produce. They can also be analyzed and evaluated by applying aesthetic, mathematical, pragmatic, and other criteria. Students in this course design and produce solutions, models, and artifacts, and they evaluate and analyze their own computational work as well as the computational work others have produced.

Students are expected to:

- Evaluate a proposed solution to a problem;
- Locate and correct errors;
- Explain how an artifact functions; and
- Justify appropriateness and correctness.

P5: Communicating

Students in this course describe computation and the impact of technology and computation, explain and justify the design and appropriateness of their computational choices, and analyze and describe both computational artifacts and the results or behaviors of such artifacts. Communication includes written and oral descriptions supported by graphs, visualizations, and computational analysis. Students are expected to:

- Explain the meaning of a result in context;
- Describe computation with accurate and precise language, notations, or visualizations; and
- Summarize the purpose of a computational artifact.

P6: Collaborating

Innovation can occur when people work together or independently. People working collaboratively can often achieve more than individuals working alone. Learning to collaborate effectively includes drawing on diverse perspectives, skills, and the backgrounds of peers to address complex and open-ended problems. Students in this course collaborate on a number of activities, including investigation of questions using data sets and in the production of computational artifacts. Students are expected to:

- Collaborate with another student in solving a computational problem;
- Collaborate with another student in producing an artifact;
- Share the workload by providing individual contributions to overall collaborative effort;
- Foster a constructive collaborative climate by resolving conflicts and facilitating the contributions of a partner or team member;
- Exchange knowledge and feedback with a partner or team member; and
- Review and revise their work as needed to create a high-quality artifact.

Concept Outline

Big Idea 1: Creativity

Computing is a creative activity. Creativity and computing are prominent forces in innovation; the innovations enabled by computing have had and will continue to have far-reaching impact. At the same time, computing facilitates exploration and the creation of computational artifacts and new knowledge that help people solve personal, societal, and global problems. This course emphasizes the creative aspects of computing. Students in this course use the tools and techniques of computer science to create interesting and relevant artifacts with characteristics that are enhanced by computation.

Essential Questions:

- How can a creative development process affect the creation of computational artifacts?
- How can computing and the use of computational tools foster creative expression?
- How can computing extend traditional forms of human expression and experience?

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
<p>1.1 Creative development can be an essential process for creating computational artifacts.</p>	<p>1.1.1 Apply a creative development process when creating computational artifacts. [P2]</p>	<p>1.1.1A A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.</p> <p>1.1.1B Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.</p>
<p>1.2 Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem.</p>	<p>1.2.1 Create a computational artifact for creative expression. [P2]</p>	<p>1.2.1A A computational artifact is anything created by a human using a computer and can be, but is not limited to, a program, an image, audio, video, a presentation, or a web page file.</p> <p>1.2.1B Creating computational artifacts requires understanding and using software tools and services.</p> <p>1.2.1C Computing tools and techniques are used to create computational artifacts and can include, but are not limited to, programming IDEs, spreadsheets, 3D printers, or text editors.</p> <p>1.2.1D A creatively developed computational artifact can be created by using nontraditional, nonprescribed computing techniques.</p>

		1.2.1E Creative expressions in a computational artifact can reflect personal expressions of ideas or interests.
1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. [P2]		1.2.2A Computing tools and techniques can enhance the process of finding a solution to a problem. 1.2.2B A creative development process for creating computational artifacts can be used to solve problems when traditional or prescribed computing techniques are not effective.
1.2.3 Create a new computational artifact by combining or modifying existing artifacts. [P2]		1.2.3A Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts. 1.2.3B Computation facilitates the creation and modification of computational artifacts with enhanced detail and precision. 1.2.3C Combining or modifying existing artifacts can show personal expression of ideas.
1.2.4 Collaborate in the creation of computational artifacts. [P6]		1.2.4A A collaboratively created computational artifact reflects effort by more than one person. 1.2.4B Effective collaborative teams consider the use of online collaborative tools. 1.2.4C Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation. 1.2.4D Effective collaboration strategies enhance performance. 1.2.4E Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts. 1.2.4F A collaboratively created computational artifact can reflect personal expressions of ideas.
1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. [P4]		1.2.5A The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact. 1.2.5B A computational artifact may have weaknesses, mistakes, or errors depending on the type of artifact. 1.2.5C The functionality of a computational artifact may be related to how it is used or perceived.

		1.2.5D The suitability (or appropriateness) of a computational artifact may be related to how it is used or perceived.
1.3 Computing can extend traditional forms of human expression and experience.	1.3.1 Use computing tools and techniques for creative expression. [P2]	<p>1.3.1A Creating digital effects, images, audio, video, and animations has transformed industries.</p> <p>1.3.1B Digital audio and music can be created by synthesizing sounds, sampling existing audio and music, and recording and manipulating sounds, including layering and looping.</p> <p>1.3.1C Digital images can be created by generating pixel patterns, manipulating existing digital images, or combining images.</p> <p>1.3.1D Digital effects and animations can be created by using existing software or modified software that includes functionality to implement the effects and animations.</p> <p>1.3.1E Computing enables creative exploration of both real and virtual phenomena.</p>

Big Idea 2: Abstraction

Abstraction reduces information and detail to facilitate focus on relevant concepts. Everyone uses abstraction on a daily basis to effectively manage complexity. In computer science, abstraction is a central problem-solving technique. It is a process, a strategy, and the result of reducing detail to focus on concepts relevant to understanding and solving problems. This course includes examples of abstractions used in modeling the world, managing complexity, and communicating with people as well as with machines. Students in this course learn to work with multiple levels of abstraction while engaging with computational problems and systems; use models and simulations that simplify complex topics in graphical, textual, and tabular formats; and use snapshots of models and simulation outputs to understand how data is changing, identify patterns, and recognize abstractions.

Essential Questions:

- How are vastly different kinds of data, physical phenomena, and mathematical concepts represented on a computer?
- How does abstraction help us in writing programs, creating computational artifacts, and solving problems?
- How can computational models and simulations help generate new understanding and knowledge?

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
--------------------------------	-------------------------------------------------------------------	-------------------------------------------------------------

<p>2.1 A variety of abstractions built upon binary sequences can be used to represent all digital data.</p>	<p>2.1.1 Describe the variety of abstractions used to represent data. [P3]</p> <p>Exclusion Statement (2.1.1E): Two's complement conversions are beyond the scope of this course and the AP Exam.</p>	<p>2.1.1A Digital data is represented by abstractions at different levels.</p> <p>2.1.1B At the lowest level, all digital data are represented by bits.</p> <p>2.1.1C At a higher level, bits are grouped to represent abstractions, including but not limited to numbers, characters, and color.</p> <p>2.1.1D Number bases, including binary, decimal, and hexadecimal, are used to represent and investigate digital data.</p> <p>2.1.1E At one of the lowest levels of abstraction, digital data is represented in binary (base 2) using only combinations of the digits zero and one.</p> <p>2.1.1F Hexadecimal (base 16) is used to represent digital data because hexadecimal representation uses fewer digits than binary.</p> <p>2.1.1G Numbers can be converted from any base to any other base.</p>
	<p>2.1.2 Explain how binary sequences are used to represent digital data. [P5]</p> <p>Exclusion Statement (2.1.2A): Binary representations of scientific notation are beyond the scope of this course and the AP Exam.</p> <p>Exclusion Statement (2.1.2B): Range limitations of any one language, compiler, or architecture are beyond the scope of this course and the AP Exam.</p>	<p>2.1.2A A finite representation is used to model the infinite mathematical concept of a number.</p> <p>2.1.2B In many programming languages, the fixed number of bits used to represent characters or integers limits the range of integer values and mathematical operations; this limitation can result in overflow or other errors.</p> <p>2.1.2C In many programming languages, the fixed number of bits used to represent real numbers (as floating-point numbers) limits the range of floating-point values and mathematical operations; this limitation can result in round-off and other errors.</p> <p>2.1.2D The interpretation of a binary sequence depends on how it is used.</p> <p>2.1.2E A sequence of bits may represent instructions or data.</p> <p>2.1.2F A sequence of bits may represent different types of data in different contexts.</p>
<p>2.2 Multiple levels of abstraction are used to write programs or create other computational</p>	<p>2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. [P2]</p> <p>Exclusion Statement</p>	<p>2.2.1A The process of developing an abstraction involves removing detail and generalizing functionality.</p> <p>2.2.1B An abstraction extracts common features from specific examples in order to generalize concepts.</p> <p>2.2.1C An abstraction generalizes functionality with</p>

artifacts.	(2.2.1C): An understanding of the difference between value and reference parameters is beyond the scope of this course and the AP Exam.	input parameters that allow software reuse.
	2.2.2 Use multiple levels of abstraction to write programs. [P3]	2.2.2A Software is developed using multiple levels of abstractions, such as constants, expressions, statements, procedures, and libraries. 2.2.2B Being aware of and using multiple levels of abstraction in developing programs helps to more effectively apply available resources and tools to solve problems.
	2.2.3 Identify multiple levels of abstractions that are used when writing programs. [P3] Exclusion Statement (2.2.3A): Knowledge of the abstraction capabilities of all programming languages is beyond the scope of this course and the AP Exam. Exclusion Statement (2.2.3F): Memorization of specific gate visual representations is beyond the scope of this course and the AP Exam.	2.2.3A Different programming languages offer different levels of abstraction. 2.2.3B High-level programming languages provide more abstractions for the programmer and make it easier for people to read and write a program. 2.2.3C Code in a programming language is often translated into code in another (lower-level) language to be executed on a computer. 2.2.3D In an abstraction hierarchy, higher levels of abstraction (the most general concepts) would be placed toward the top and lower-level abstractions (the more specific concepts) toward the bottom. 2.2.3E Binary data is processed by physical layers of computing hardware, including gates, chips, and components. 2.2.3F A logic gate is a hardware abstraction that is modeled by a Boolean function. 2.2.3G A chip is an abstraction composed of low-level components and circuits that perform a specific function. 2.2.3H A hardware component can be low level like a transistor or high level like a video card. 2.2.3I Hardware is built using multiple levels of abstractions, such as transistors, logic gates, chips, memory, motherboards, special purposes cards, and storage devices. 2.2.3J Applications and systems are designed, developed, and analyzed using levels of hardware, software, and conceptual abstractions. 2.2.3K Lower-level abstractions can be combined to make higher-level abstractions, such as short

		message services (SMS) or email messages, images, audio files, and videos.
2.3 Models and simulations use abstraction to generate new understanding and knowledge.	2.3.1 Use models and simulations to represent phenomena. [P3]	<p>2.3.1A Models and simulations are simplified representations of more complex objects or phenomena.</p> <p>2.3.1B Models may use different abstractions or levels of abstraction depending on the objects or phenomena being posed.</p> <p>2.3.1C Models often omit unnecessary features of the objects or phenomena that are being modeled.</p> <p>2.3.1D Simulations mimic real-world events without the cost or danger of building and testing the phenomena in the real world.</p>
	2.3.2 Use models and simulations to formulate, refine, and test hypotheses. [P3]	<p>2.3.2A Models and simulations facilitate the formulation and refinement of hypotheses related to the objects or phenomena under consideration.</p> <p>2.3.2B Hypotheses are formulated to explain the objects or phenomena being modeled.</p> <p>2.3.2C Hypotheses are refined by examining the insights that models and simulations provide into the objects or phenomena.</p> <p>2.3.2D The results of simulations may generate new knowledge and new hypotheses related to the phenomena being modeled.</p> <p>2.3.2E Simulations allow hypotheses to be tested without the constraints of the real world.</p> <p>2.3.2F Simulations can facilitate extensive and rapid testing of models.</p> <p>2.3.2G The time required for simulations is impacted by the level of detail and quality of the models, and the software and hardware used for the simulation.</p> <p>2.3.2H Rapid and extensive testing allows models to be changed to accurately reflect the objects or phenomena being modeled.</p>

Big Idea 3: Data and Information

Data and information facilitate the creation of knowledge. Computing enables and empowers new methods of information processing, driving monumental change across many disciplines — from art to business to science. Managing and interpreting an overwhelming amount of raw data is part of the

foundation of our information society and economy. People use computers and computation to translate, process, and visualize raw data, and to create information. Computation and computer science facilitate and enable a new understanding of data and information that contributes knowledge to the world. Students in this course work with data using a variety of computational tools and techniques to better understand the many ways in which data is transformed into information and knowledge.

Essential Questions:

- How can computation be employed to help people process data and information to gain insight and knowledge?
- How can computation be employed to facilitate exploration and discovery when working with data?
- What considerations and trade-offs arise in the computational manipulation of data?
- What opportunities do large data sets provide for solving problems and creating knowledge?

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
3.1 People use computer programs to process information to gain insight and knowledge.	3.1.1 Use computers to process information, find patterns, and test hypotheses about digitally processed information to gain insight and knowledge. [P4]	3.1.1A Computers are used in an iterative and interactive way when processing digital information to gain insight and knowledge. 3.1.1B Digital information can be filtered and cleaned by using computers to process information. 3.1.1C Combining data sources, clustering data, and data classification are part of the process of using computers to process information. 3.1.1D Insight and knowledge can be obtained from translating and transforming digitally represented information. 3.1.1E Patterns can emerge when data is transformed using computational tools.
	3.1.2 Collaborate when processing information to gain insight and knowledge. [P6]	3.1.2A Collaboration is an important part of solving data-driven problems. 3.1.2B Collaboration facilitates solving computational problems by applying multiple perspectives, experiences, and skill sets. 3.1.2C Communication between participants working on data-driven problems gives rise to enhanced insights and knowledge. 3.1.2D Collaboration in developing hypotheses and questions, and in testing hypotheses and answering questions, about data helps participants gain insight and knowledge. 3.1.2E Collaborating face-to-face and using online collaborative tools can facilitate processing

		<p>information to gain insight and knowledge.</p> <p>3.1.2F Investigating large data sets collaboratively can lead to insight and knowledge not obtained when working alone.</p>
	<p>3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notations, and precise language. [P5]</p>	<p>3.1.3A Visualization tools and software can communicate information about data.</p> <p>3.1.3B Tables, diagrams, and textual displays can be used in communicating insight and knowledge gained from data.</p> <p>3.1.3C Summaries of data analyzed computationally can be effective in communicating insight and knowledge gained from digitally represented information.</p> <p>3.1.3D Transforming information can be effective in communicating knowledge gained from data.</p> <p>3.1.3E Interactivity with data is an aspect of communicating.</p>
<p>3.2 Computing facilitates exploration and the discovery of connections in information.</p>	<p>3.2.1 Extract information from data to discover and explain connections, patterns, or trends. [P1]</p> <p>Exclusion Statement (3.2.1F): Students are not expected to know specific formulas or options available in spreadsheet or database software packages..</p>	<p>3.2.1A Large data sets provide opportunities and challenges for extracting information and knowledge.</p> <p>3.2.1B Large data sets provide opportunities for identifying trends, making connections in data, and solving problems.</p> <p>3.2.1C Computing tools facilitate the discovery of connections in information within large data sets.</p> <p>3.2.1D Search tools are essential for efficiently finding information.</p> <p>3.2.1E Information filtering systems are important tools for finding information and recognizing patterns in the information.</p> <p>3.2.1F Software tools, including spreadsheets and databases, help to efficiently organize and find trends in information.</p> <p>3.2.1G Metadata is data about data.</p> <p>3.2.1H Metadata can be descriptive data about an image, a Web page, or other complex objects.</p> <p>3.2.1I Metadata can increase the effective use of data or data sets by providing additional information about various aspects of that data.</p>
	<p>3.2.2. Use large data sets to explore and discover information and knowledge. [P3]</p>	<p>3.2.2A Large data sets include data such as transactions, measurements, text, sound, images, and video.</p> <p>3.2.2B The storing, processing, and curating of large</p>

		<p>data sets is challenging.</p> <p>3.2.2C Structuring large data sets for analysis can be challenging.</p> <p>3.2.2D Maintaining privacy of large data sets containing personal information can be challenging.</p> <p>3.2.2E Scalability of systems is an important consideration when data sets are large.</p> <p>3.2.2F The size or scale of a system that stores data affects how that data set is used.</p> <p>3.2.2G The effective use of large data sets requires computational solutions.</p> <p>3.2.2H Analytical techniques to store, manage, transmit, and process data sets change as the size of data sets scale.</p>
<p>3.3 There are trade-offs when representing information as digital data.</p>	<p>3.3.1 Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information. [P4]</p>	<p>3.3.1A Digital data representations involve trade-offs related to storage, security, and privacy concerns.</p> <p>3.3.1B Security concerns engender trade-offs in storing and transmitting information.</p> <p>3.3.1C There are trade-offs in using lossy and lossless compression techniques for storing and transmitting data.</p> <p>3.3.1D Lossless data compression reduces the number of bits stored or transmitted but allows complete reconstruction of the original data.</p> <p>3.3.1E Lossy data compression can significantly reduce the number of bits stored or transmitted at the cost of being able to reconstruct only an approximation of the original data.</p> <p>3.3.1F Security and privacy concerns arise with data containing personal information.</p> <p>3.3.1G Data is stored in many formats depending on its characteristics (e.g., size and intended use).</p> <p>3.3.1H The choice of storage media affects both the methods and costs of manipulating the data it contains.</p> <p>3.3.1I Reading data and updating data have different storage requirements.</p>

Big Idea 4: Algorithms

Algorithms are used to develop and express solutions to computational problems. Algorithms are fundamental to even the most basic everyday task. Algorithms realized in software have affected the world in profound and lasting ways. Secure data transmission and quick access to large amounts of relevant information are made possible through the implementation of algorithms. The development, use, and analysis of algorithms are some of the most fundamental aspects of computing. Students in this course work with algorithms in many ways: they develop and express original algorithms, they implement algorithms in a language, and they analyze algorithms analytically and empirically.

Essential Questions:

- How are algorithms implemented and executed on computers and computational devices?
- Why are some languages better than others when used to implement algorithms?
- What kinds of problems are easy, what kinds are difficult, and what kinds are impossible to solve algorithmically?
- How are algorithms evaluated?

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
4.1 Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages.	4.1.1 Develop an algorithm for implementation in a program. [P2]	4.1.1A Sequencing, selection, and iteration are building blocks of algorithms. 4.1.1B Sequencing is the application of each step of an algorithm in the order in which the statements are given. 4.1.1C Selection uses a Boolean condition to determine which of two parts of an algorithm is used. 4.1.1D Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times. 4.1.1E Algorithms can be combined to make new algorithms. 4.1.1F Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct. 4.1.1G Knowledge of standard algorithms can help in constructing new algorithms. 4.1.1H Different algorithms can be developed to solve the same problem. 4.1.1I Developing a new algorithm to solve a problem can yield insight into the problem.
	4.1.2 Express an algorithm	4.1.2A Languages for algorithms include natural

	<p>in a language. [P5]</p>	<p>language, pseudocode, and visual and textual programming languages.</p> <p>4.1.2B Natural language and pseudocode describe algorithms so that humans can understand them.</p> <p>4.1.2C Algorithms described in programming languages can be executed on a computer.</p> <p>4.1.2D Different languages are better suited for expressing different algorithms.</p> <p>4.1.2E Some programming languages are designed for specific domains and are better for expressing algorithms in those domains.</p> <p>4.1.2F The language used to express an algorithm can affect characteristics such as clarity or readability but not whether an algorithmic solution exists.</p> <p>4.1.2G Every algorithm can be constructed using only sequencing, selection, and iteration.</p> <p>4.1.2H Nearly all programming languages are equivalent in terms of being able to express any algorithm.</p> <p>4.1.2I Clarity and readability are important considerations when expressing an algorithm in a language.</p>
<p>4.2 Algorithms can solve many but not all computational problems.</p>	<p>4.2.1 Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time. [P1]</p> <p>Exclusion Statement (LO 4.2.1): Any discussion of nondeterministic polynomial (NP) is beyond the scope of this course and the AP Exam.</p>	<p>4.2.1A Many problems can be solved in a reasonable time.</p> <p>4.2.1B Reasonable time means that as the input size grows, the number of steps the algorithm takes is proportional to the square (or cube, fourth power, fifth power, etc.) of the size of the input.</p> <p>4.2.1C Some problems cannot be solved in a reasonable time, even for small input sizes.</p> <p>4.2.1D Some problems can be solved but not in a reasonable time. In these cases, heuristic approaches may be helpful to find solutions in reasonable time.</p>
	<p>4.2.2 Explain the difference between solvable and unsolvable problems in computer science. [P1]</p> <p>Exclusion Statement (LO 4.2.2): Determining whether a given problem is</p>	<p>4.2.2A A heuristic is a technique that may allow us to find an approximate solution when typical methods fail to find an exact solution.</p> <p>4.2.2B Heuristics may be helpful for finding an approximate solution more quickly when exact methods are too slow.</p> <p>4.2.2C Some optimization problems such as “find the best” or “find the smallest” cannot be solved in</p>

	<p>solvable or unsolvable is beyond the scope of this course and the AP Exam.</p> <p>Exclusion Statement (4.2.2B): Specific heuristic solutions are beyond the scope of this course and the AP Exam.</p>	<p>a reasonable time, but approximations to the optimal solution can.</p> <p>4.2.2D Some problems cannot be solved using any algorithm.</p>
	<p>4.2.3 Explain the existence of undecidable problems in computer science. [P1]</p> <p>Exclusion Statement (4.2.3C): Determining whether a given problem is undecidable is beyond the scope of this course and the AP Exam.</p>	<p>4.2.3A An undecidable problem may have instances that have an algorithmic solution, but there is no algorithmic solution that solves all instances of the problem.</p> <p>4.2.3B A decidable problem is one in which an algorithm can be constructed to answer “yes” or “no” for all inputs (e.g., “is the number even?”).</p> <p>4.2.3C An undecidable problem is one in which no algorithm can be constructed that always leads to a correct yes-or-no answer.</p>
	<p>4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity. [P4]</p> <p>Exclusion statement (4.2.4C): Formally proving program correctness is beyond the scope of this course and the AP Exam.</p> <p>Exclusion Statement (4.2.4G): Formal analysis of algorithms (Big-O) and formal reasoning using mathematical formulas are beyond the scope of this course and the AP Exam.</p>	<p>4.2.4A Determining an algorithm’s efficiency is done by reasoning formally or mathematically about the algorithm.</p> <p>4.2.4B Empirical analysis of an algorithm is done by implementing the algorithm and running it on different inputs.</p> <p>4.2.4C The correctness of an algorithm is determined by reasoning formally or mathematically about the algorithm, not by testing an implementation of the algorithm.</p> <p>4.2.4D Different correct algorithms for the same problem can have different efficiencies.</p> <p>4.2.4E Sometimes more efficient algorithms are more complex.</p> <p>4.2.4F Finding an efficient algorithm for a problem can help solve larger instances of the problem.</p> <p>4.2.4G Efficiency includes both execution time and memory usage.</p> <p>4.2.4H Linear search can be used when searching for an item in any list; binary search can be used only when the list is sorted.</p>

Big Idea 5: Programming

Programming enables problem solving, human expression, and creation of knowledge.

Programming and the creation of software has changed our lives. Programming results in the creation of software, and it facilitates the creation of computational artifacts, including music, images, and visualizations. In this course, programming enables exploration and is the object of study. This course introduces students to the concepts and techniques related to writing programs, developing software, and using software effectively; the focus of the course is not on programming per se but on all aspects of computation. Students in this course create programs, translating human intention into computational artifacts.

Essential Questions:

- How are programs developed to help people, organizations, or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity, or to create new knowledge?
- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
<p>5.1 Programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems (to help people, organizations, or society).</p>	<p>5.1.1 Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge. [P2]</p>	<p>5.1.1A Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.</p> <p>5.1.1B Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.</p> <p>5.1.1C Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may be developed with different standards or methods than programs developed for widespread distribution.</p> <p>5.1.1D Additional desired outcomes may be realized independently of the original purpose of the program.</p> <p>5.1.1E A computer program or the results of running a program may be rapidly shared with a large number of users and can have widespread impact on individuals, organizations, and society.</p> <p>5.1.1F Advances in computing have generated and</p>

		increased creativity in other fields.
	<p>5.1.2 Develop a correct program to solve problems. [P2]</p>	<p>5.1.2A An iterative process of program development helps in developing a correct program to solve problems.</p> <p>5.1.2B Developing correct program components and then combining them helps in creating correct programs.</p> <p>5.1.2C Incrementally adding tested program segments to correct, working programs helps create large correct programs.</p> <p>5.1.2D Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.</p> <p>5.1.2E Documentation about program components, such as blocks and procedures, helps in developing and maintaining programs.</p> <p>5.1.2F Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments.</p> <p>5.1.2G Program development includes identifying programmer and user concerns that affect the solution to problems.</p> <p>5.1.2H Consultation and communication with program users is an important aspect of program development to solve problems.</p> <p>5.1.2I A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.</p> <p>5.1.2J A programmer designs, implements, tests, debugs, and maintains programs when solving problems.</p>
	<p>5.1.3 Collaborate to develop a program. [P6]</p>	<p>5.1.3A Collaboration can decrease the size and complexity of tasks required of individual programmers.</p> <p>5.1.3B Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming.</p> <p>5.1.3C Collaboration in the iterative development of a program requires different skills than developing a program alone.</p> <p>5.1.3D Collaboration can make it easier to find and correct errors when developing programs.</p> <p>5.1.3E Collaboration facilitates developing program components independently.</p>

		<p>5.1.3F Effective communication between participants is required for successful collaboration when developing programs.</p>
<p>5.2 People write programs to execute algorithms.</p>	<p>5.2.1 Explain how programs implement algorithms. [P3]</p>	<p>5.2.1A Algorithms are implemented using program instructions that are processed during program execution.</p> <p>5.2.1B Program instructions are executed sequentially.</p> <p>5.2.1C Program instructions may involve variables that are initialized and updated, read, and written.</p> <p>5.2.1D An understanding of instruction processing and program execution is useful for programming.</p> <p>5.2.1E Program execution automates processes.</p> <p>5.2.1F Processes use memory, a central processing unit (CPU), and input and output.</p> <p>5.2.1G A process may execute by itself or with other processes.</p> <p>5.2.1H A process may execute on one or several CPUs.</p> <p>5.2.1I Executable programs increase the scale of problems that can be addressed.</p> <p>5.2.1J Simple algorithms can solve a large set of problems when automated.</p> <p>5.2.1K Improvements in algorithms, hardware, and software increase the kinds of problems and the size of problems solvable by programming.</p>
<p>5.3 Programming is facilitated by appropriate abstractions.</p>	<p>5.3.1 Use abstraction to manage complexity in programs. [P3]</p>	<p>5.3.1A Procedures are reusable programming abstractions.</p> <p>5.3.1B A function is a named grouping of programming instructions.</p> <p>5.3.1C Procedures reduce the complexity of writing and maintaining programs.</p> <p>5.3.1D Procedures have names and may have parameters and return values.</p> <p>5.3.1E Parameterization can generalize a specific solution.</p> <p>5.3.1F Parameters generalize a solution by allowing a function to be used instead of duplicated code.</p> <p>5.3.1G Parameters provide different values as input to procedures when they are called in a program.</p> <p>5.3.1H Data abstraction provides a means of separating behavior from implementation.</p> <p>5.3.1I Strings and string operations, including concatenation and some form of substring, are</p>

		<p>common in many programs.</p> <p>5.3.1J Integers and floating-point numbers are used in programs without requiring understanding of how they are implemented.</p> <p>5.3.1K Lists and list operations, such as add, remove, and search, are common in many programs.</p> <p>5.3.1L Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.</p> <p>5.3.1M Application program interfaces (APIs) and libraries simplify complex programming tasks.</p> <p>5.3.1N Documentation for an API/library is an important aspect of programming.</p> <p>5.3.1O APIs connect software components, allowing them to communicate.</p>
<p>5.4 Programs are developed, maintained, and used by people for different purposes.</p>	<p>5.4.1 Evaluate the correctness of a program. [P4]</p>	<p>5.4.1A Program style can affect the determination of program correctness.</p> <p>5.4.1B Duplicated code can make it harder to reason about a program.</p> <p>5.4.1C Meaningful names for variables and procedures help people better understand programs.</p> <p>5.4.1D Longer code blocks are harder to reason about than shorter code blocks in a program.</p> <p>5.4.1E Locating and correcting errors in a program is called debugging the program.</p> <p>5.4.1F Knowledge of what a program is supposed to do is required in order to find most program errors.</p> <p>5.4.1G Examples of intended behavior on specific inputs help people understand what a program is supposed to do.</p> <p>5.4.1H Visual displays (or different modalities) of program state can help in finding errors.</p> <p>5.4.1I Programmers justify and explain a program's correctness.</p> <p>5.4.1J Justification can include a written explanation about how a program meets its specifications.</p> <p>5.4.1K Correctness of a program depends on correctness of program components, including code blocks and procedures.</p> <p>5.4.1L An explanation of a program helps people understand the functionality and purpose of it.</p> <p>5.4.1M The functionality of a program is often</p>

		<p>described by how a user interacts with it.</p> <p>5.4.1N The functionality of a program is best described at a high level by what the program does, not at the lower level of how the program statements work to accomplish this.</p>
<p>5.5 Programming uses mathematical and logical concepts.</p>	<p>5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1]</p> <p>Exclusion Statement (5.5.1B): Specific range limitations of all programming languages are beyond the scope of this course and the AP Exam.</p> <p>Exclusion Statement (5.5.1C): Specific sets of values that cannot be exactly represented by floating point numbers are beyond the scope of this course and the AP Exam.</p>	<p>5.5.1A Numbers and numerical concepts are fundamental to programming.</p> <p>5.5.1B Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.</p> <p>5.5.1C Real numbers are approximated by floating-point representations that do not necessarily have infinite precision.</p> <p>5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.</p> <p>5.5.1E Logical concepts and Boolean algebra are fundamental to programming.</p> <p>5.5.1F Compound expressions using <i>and</i>, <i>or</i>, and <i>not</i> are part of most programming languages.</p> <p>5.5.1G Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.</p> <p>5.5.1H Computational methods may use lists and collections to solve problems.</p> <p>5.5.1I Lists and other collections can be treated as abstract data types (ADTs) in developing programs.</p> <p>5.5.1J Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection.</p>

Big Idea 6: The Internet

The Internet pervades modern computing. The Internet and the systems built on it have had a profound impact on society. Computer networks support communication and collaboration. The principles of systems and networks that helped enable the Internet are also critical in the implementation of computational solutions. Students in this course gain insight into how the Internet operates, study characteristics of the Internet and systems built upon it, and analyze important concerns such as cybersecurity.

Essential Questions:

- What is the Internet? How is it built? How does it function?

- What aspects of the Internet’s design and development have helped it scale and flourish?
- How is cybersecurity impacting the ever-increasing number of Internet users?

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
6.1 The Internet is a network of autonomous systems.	<p>6.1.1 Explain the abstractions in the Internet and how the Internet functions. [P3]</p> <p>Exclusion Statement (LO 6.1.1): Specific devices used to implement the abstractions in the Internet are beyond the scope of this course and the AP Exam.</p> <p>Exclusion Statement (6.1.1F): Specific details of any particular standard for addresses are beyond the scope of this course and the AP Exam.</p>	<p>6.1.1A The Internet connects devices and networks all over the world.</p> <p>6.1.1B An end-to-end architecture facilitates connecting new devices and networks on the Internet.</p> <p>6.1.1C Devices and networks that make up the Internet are connected and communicate using addresses and protocols.</p> <p>6.1.1D The Internet and the systems built on it facilitate collaboration.</p> <p>6.1.1E Connecting new devices to the Internet is enabled by assignment of an Internet protocol (IP) address.</p> <p>6.1.1F The Internet is built on evolving standards, including those for addresses and names.</p> <p>6.1.1G The domain name system (DNS) translates names to IP addresses.</p> <p>6.1.1H The number of devices that could use an IP address has grown so fast that a new protocol (IPv6) has been established to handle routing of many more devices.</p> <p>6.1.1I Standards such as hypertext transfer protocol (HTTP), IP, and simple mail transfer protocol (SMTP) are developed and overseen by the Internet Engineering Task Force (IETF).</p>
6.2 Characteristics of the Internet influence the systems built on it.	6.2.1 Explain characteristics of the Internet and the systems built on it. [P5]	<p>6.2.1A The Internet and the systems built on it are hierarchical and redundant.</p> <p>6.2.1B The domain name syntax is hierarchical.</p> <p>6.2.1C IP addresses are hierarchical.</p> <p>6.2.1D Routing on the Internet is fault tolerant and redundant.</p>
	6.2.2 Explain how the characteristics of the Internet influence the systems built on it. [P4]	<p>6.2.2A Hierarchy and redundancy help systems scale.</p> <p>6.2.2B The redundancy of routing (i.e., more than one way to route data) between two points on the Internet increases the reliability of the Internet and helps it scale to more devices and more people.</p>

	<p>Exclusion Statements (6.2.2F): Specific details of any particular packet-switching system are beyond the scope of this course and the AP Exam.</p> <p>Exclusion Statement (6.2.2G): Specific technical details of how TCP/IP works are beyond the scope of this course and the AP Exam.</p> <p>Exclusion Statement (6.2.2H): Understanding the technical aspects of how SSL/TLS works is beyond the scope of this course and the AP Exam.</p>	<p>6.2.2C Hierarchy in the DNS helps that system scale.</p> <p>6.2.2D Interfaces and protocols enable widespread use of the Internet.</p> <p>6.2.2E Open standards fuel the growth of the Internet.</p> <p>6.2.2F The Internet is a packet-switched system through which digital data is sent by breaking the data into blocks of bits called packets, which contain both the data being transmitted and control information for routing the data.</p> <p>6.2.2G Standards for packets and routing include transmission control protocol/Internet protocol (TCP/IP).</p> <p>6.2.2H Standards for sharing information and communicating between browsers and servers on the Web include HTTP and secure sockets layer/transport layer security (SSL/TLS).</p> <p>6.2.2I The size and speed of systems affect their use.</p> <p>6.2.2J The bandwidth of a system is a measure of bit rate — the amount of data (measured in bits) that can be sent in a fixed amount of time.</p> <p>6.2.2K The latency of a system is the time elapsed between the transmission and the receipt of a request.</p>
<p>6.3 Cybersecurity is an important concern for the Internet and the systems built on it.</p>	<p>6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it. [P1]</p> <p>Exclusion statement (6.3.1I): Specific mathematical functions used in cryptography are</p>	<p>6.3.1A The trust model of the Internet involves trade-offs.</p> <p>6.3.1B The domain name system (DNS) was not designed to be completely secure.</p> <p>6.3.1C Implementing cybersecurity has software, hardware, and human components.</p> <p>6.3.1D Cyber warfare and cyber crime have widespread and potentially devastating effects.</p> <p>6.3.1E Distributed denial-of-service attacks (DDoS) compromise a target by flooding it with requests from multiple systems.</p> <p>6.3.1F Phishing, viruses, and other attacks have human and software components.</p> <p>6.3.1G Antivirus software and firewalls can help prevent unauthorized access to private data.</p> <p>6.3.1H Cryptography is essential to many models of cybersecurity.</p> <p>6.3.1I Cryptography has a mathematical foundation.</p>

	<p>beyond the scope of this course and the AP Exam.</p> <p>Exclusion Statement (6.3.1K): The methods used in encryption are beyond the scope of this course and the AP Exam.</p> <p>Exclusion statement (6.3.1L): The mathematical methods used in public key encryption are beyond the scope of this course and the AP Exam.</p> <p>Exclusion statement (6.3.1M): The technical details of the process certificate authorities follow are beyond the scope of this course and the AP Exam.</p>	<p>6.3.1J Open standards help ensure cryptography is secure.</p> <p>6.3.1K Symmetric encryption is a method of encryption involving one key for encryption and decryption.</p> <p>6.3.1L Public key encryption, which is not symmetric, is an encryption method that is widely used because of the enhanced security associated with its use.</p> <p>6.3.1M Certificate authorities (CAs) issue digital certificates that validate the ownership of encrypted keys used in secured communication and are based on a trust model.</p>
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Big Idea 7: Global Impact

Computing has global impact. Computation has changed the way people think, work, live, and play. Our methods for communicating, collaborating, problem solving, and doing business have changed and are changing due to innovations enabled by computing. Many innovations in other fields are fostered by advances in computing. Computational approaches lead to new understandings, new discoveries, and new disciplines. Students in this course become familiar with many ways in which computing enables innovation, and they analyze the potential benefits and harmful effects of computing in a number of contexts.

Essential Questions:

- How does computing enhance human communication, interaction, and cognition?
- How does computing enable innovation?
- What are some potential beneficial and harmful effects of computing?
- How do economic, social, and cultural contexts influence innovation and the use of computing?

<p>Enduring Understandings</p>	<p>Learning Objectives (What students must</p>	<p>Essential Knowledge (What students need to know)</p>
---------------------------------------	-------------------------------------------------------	----------------------------------------------------------------

	be able to do)	
7.1 Computing enhances communication, interaction, and cognition.	<p>7.1.1 Explain how computing innovations affect communication, interaction, and cognition. [P4]</p> <p>Exclusion Statement (7.1.1C): Detailed knowledge of any social media site is beyond the scope of this course and the AP Exam.</p>	<p>7.1.1A Email, short message service (SMS), and chat have fostered new ways to communicate and collaborate.</p> <p>7.1.1B Video conferencing and video chat have fostered new ways to communicate and collaborate.</p> <p>7.1.1C Social media continues to evolve and foster new ways to communicate.</p> <p>7.1.1D Cloud computing fosters new ways to communicate and collaborate.</p> <p>7.1.1E Widespread access to information facilitates the identification of problems, development of solutions, and dissemination of results.</p> <p>7.1.1F Public data provides widespread access and enables solutions to identified problems.</p> <p>7.1.1G Search trends are predictors.</p> <p>7.1.1H Social media, such as blogs and Twitter, have enhanced dissemination.</p> <p>7.1.1I Global Positioning System (GPS) and related technologies have changed how humans travel, navigate, and find information related to geolocation.</p> <p>7.1.1J Sensor networks facilitate new ways of interacting with the environment and with physical systems.</p> <p>7.1.1K Smart grids, smart buildings, and smart transportation are changing and facilitating human capabilities.</p> <p>7.1.1L Computing contributes to many assistive technologies that enhance human capabilities.</p> <p>7.1.1M The Internet and the Web have enhanced methods of and opportunities for communication and collaboration.</p> <p>7.1.1N The Internet and the Web have changed many areas, including e-commerce, health care, access to information and entertainment, and online learning.</p> <p>7.1.1O The Internet and the Web have impacted productivity, positively and negatively, in many areas.</p>
	<p>7.1.2 Explain how people participate in a problem-solving process that scales.</p>	<p>7.1.2A Distributed solutions must scale to solve some problems.</p> <p>7.1.2B Science has been impacted by using scale</p>

	[P4]	<p>and “citizen science” to solve scientific problems using home computers in scientific research.</p> <p>7.1.2C Human computation harnesses contributions from many humans to solve problems related to digital data and the Web.</p> <p>7.1.2D Human capabilities are enhanced by digitally enabled collaboration.</p> <p>7.1.2E Some online services use the contributions of many people to benefit both individuals and society.</p> <p>7.1.2F Crowdsourcing offers new models for collaboration, such as connecting people with jobs and businesses with funding.</p> <p>7.1.2G The move from desktop computers to a proliferation of always-on mobile computers is leading to new applications.</p>
7.2 Computing enables innovation in nearly every field.	7.2.1 Explain how computing has impacted innovations in other fields. [P1]	<p>7.2.1A Machine learning and data mining have enabled innovation in medicine, business, and science.</p> <p>7.2.1B Scientific computing has enabled innovation in science and business.</p> <p>7.2.1C Computing enables innovation by providing access to and sharing of information.</p> <p>7.2.1D Open access and Creative Commons have enabled broad access to digital information.</p> <p>7.2.1E Open and curated scientific databases have benefited scientific researchers.</p> <p>7.2.1F Moore’s law has encouraged industries that use computers to effectively plan future research and development based on anticipated increases in computing power.</p> <p>7.2.1G Advances in computing as an enabling technology have generated and increased the creativity in other fields.</p>
7.3 Computing has a global affect — both beneficial and harmful — on people and society.	7.3.1 Analyze the beneficial and harmful effects of computing. [P4]	<p>7.3.1A Innovations enabled by computing raise legal and ethical concerns.</p> <p>7.3.1B Commercial access to music and movie downloads and streaming raises legal and ethical concerns.</p> <p>7.3.1C Access to digital content via peer-to-peer networks raises legal and ethical concerns.</p> <p>7.3.1D Both authenticated and anonymous access to digital information raise legal and ethical concerns.</p>

		<p>7.3.1E Commercial and governmental censorship of digital information raise legal and ethical concerns.</p> <p>7.3.1F Open source and licensing of software and content raise legal and ethical concerns.</p> <p>7.3.1G Privacy and security concerns arise in the development and use of computational systems and artifacts.</p> <p>7.3.1H Aggregation of information, such as geolocation, cookies, and browsing history, raises privacy and security concerns.</p> <p>7.3.1I Anonymity in online interactions can be enabled through the use of online anonymity software and proxy servers.</p> <p>7.3.1J Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions.</p> <p>7.3.1K People can have instant access to vast amounts of information online; accessing this information can enable the collection of both individual and aggregate data that can be used and collected.</p> <p>7.3.1L Commercial and governmental curation of information may be exploited if privacy and other protections are ignored.</p> <p>7.3.1M Targeted advertising is used to help individuals, but it can be misused at both individual and aggregate levels.</p> <p>7.3.1N Widespread access to digitized information raises questions about intellectual property.</p> <p>7.3.1O Creation of digital audio, video, and textual content by combining existing content has been impacted by copyright concerns.</p> <p>7.3.1P The Digital Millennium Copyright Act (DMCA) has been a benefit and a challenge in making copyrighted digital material widely available.</p> <p>7.3.1Q Open source and free software have practical, business, and ethical impacts on widespread access to programs, libraries, and code.</p>
<p>7.4 Computing innovations influence and are influenced by the economic, social, and cultural</p>	<p>7.4.1 Explain the connections between computing and economic, social, and cultural contexts. [P1]</p>	<p>7.4.1A The innovation and impact of social media and online access is different in different countries and in different socioeconomic groups.</p> <p>7.4.1B Mobile, wireless, and networked computing have an impact on innovation throughout the world.</p>

<p>contexts in which they are designed and used.</p>		<p>7.4.1C The global distribution of computing resources raises issues of equity, access, and power.</p> <p>7.4.1D Groups and individuals are affected by the “digital divide” — differing access to computing and the Internet based on socioeconomic or geographic characteristics.</p> <p>7.4.1E Networks and infrastructure are supported by both commercial and governmental initiatives.</p>
-------------------------------------------------------------	--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Contributors:

Don Allen, Troy High School
Christine Alvarado, University of California, San Diego
Owen Astrachan, Duke University
Stacey Armstrong, Cypress Woods High School
Duane Bailey, Williams College
Tiffany Barnes, University of North Carolina at Charlotte
Charmaine Bentley, Franklin D. Roosevelt High School
Amy Briggs, Middlebury College
Gail Chapman, Computer Science Teachers Association
Tom Cortina, Carnegie Mellon University
Stephen Edwards, Virginia Tech
Dan Garcia, University of California, Berkeley*
Christina Gardner-McCune, Clemson University*
Joanna Goode, University of Oregon
Mark Guzdial, Georgia Tech
Susanne Hambruch, Purdue University
Michelle Hutton, Computer Science Teachers Association
Rich Kick, Newbury Park High School*
Andrew Kuemmel, West Madison High School*
Deepak Kumar, Bryn Mawr College
James Kurose, University of Massachusetts Amherst
Andrea Lawrence, Spellman College
Deepa Muralidhar, North Gwinnett High School*
Richard Pattis, University of California, Irvine
Jody Paul, Metropolitan State University of Denver
Dale Reed, University of Illinois at Chicago*
Eric Roberts, Stanford University
Katie Siek, University of Colorado Boulder
Beth Simon, University of California, San Diego
Larry Snyder, University of Washington
Chris Stephenson, Computer Science Teachers Association
Lynn Andrea Stein, Olin College
Fran Tress, Rutgers University*
Cameron Wilson, Association for Computing Machinery

*Members of the AP Computer Science Principles Development Committee