AP® Computer Science Principles

2016–2017
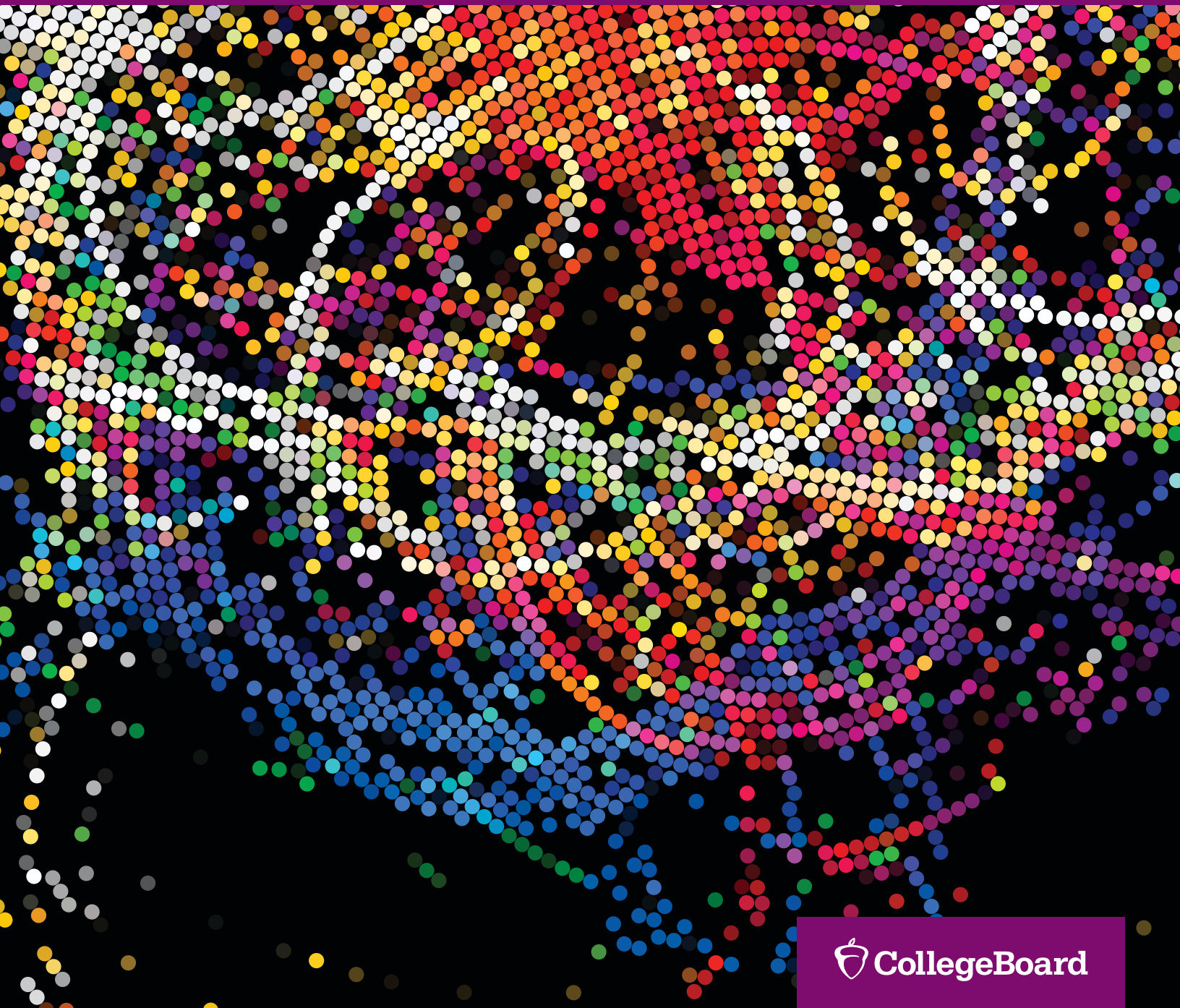
CollegeBoard

AP® Computer Science Principles

Curriculum Framework

2016–2017

CollegeBoard

New York, NY

About the College Board

The College Board is a mission-driven not-for-profit organization that connects students to college success and opportunity. Founded in 1900, the College Board was created to expand access to higher education. Today, the membership association is made up of over 6,000 of the world's leading educational institutions and is dedicated to promoting excellence and equity in education. Each year, the College Board helps more than seven million students prepare for a successful transition to college through programs and services in college readiness and college success — including the SAT® and the Advanced Placement Program®. The organization also serves the education community through research and advocacy on behalf of students, educators, and schools. For further information, visit www.collegeboard.org.

AP® Equity and Access Policy

The College Board strongly encourages educators to make equitable access a guiding principle for their AP programs by giving all willing and academically prepared students the opportunity to participate in AP. We encourage the elimination of barriers that restrict access to AP for students from ethnic, racial, and socioeconomic groups that have been traditionally underserved. Schools should make every effort to ensure their AP classes reflect the diversity of their student population. The College Board also believes that all students should have access to academically challenging course work before they enroll in AP classes, which can prepare them for AP success. It is only through a commitment to equitable preparation and access that true equity and excellence can be achieved.

The *AP Computer Science Principles Curriculum Framework* is designed to provide educators with a first look at essential information needed to understand the design and intent of the AP Computer Science Principles course in advance of its implementation in schools in the 2016-2017 academic year. Please be advised that the information contained in this publication is subject to change. The final course and exam information will be available in the *AP Computer Science Principles Course and Exam Description*, which will be published in early 2016.

Contents

Acknowledgments

The College Board would like to acknowledge the following committee members, consultants, and reviewers for their assistance with and commitment to the development of this curriculum:

AP Curriculum and Content Development Senior Director for AP Computer Science Principles

Lien Diaz, *Senior Director, AP Curriculum and Content Development*

Introduction

AP® Computer Science Principles introduces students to the central ideas of computer science, instilling the ideas and practices of computational thinking and inviting students to understand how computing changes the world. The rigorous course promotes deep learning of computational content, develops computational thinking skills, and engages students in the creative aspects of the field.

The course is unique in its focus on fostering students to **be creative**. Students are encouraged to apply creative processes when developing computational artifacts and to think creatively while using simulations to explore questions that interest them. Rather than teaching a particular programming language or tool, the course focuses on using technology and programming as a means to solve computational problems and create exciting and personally relevant artifacts. Students design and implement innovative solutions using an iterative process similar to what artists, writers, computer scientists, and engineers use to bring ideas to life.

To appeal to a broader audience, including those often underrepresented in computing, this course highlights the relevance of computer science by emphasizing the vital impact advances in computing have on people and society. By focusing the course beyond the study of machines and systems, students also have the opportunity to investigate the innovations in other fields that computing has made possible and examine the ethical implications of new computing technologies.

Students who take an AP Computer Science Principles course using this curriculum framework as its foundation will develop a range of skills vital to success in subsequent college courses, such as using computational tools to analyze and study data and working with large data sets to analyze, visualize, and draw conclusions from trends. They will also develop effective communication and collaboration skills, working individually and collaboratively to solve problems, and discussing and writing about the importance of these problems and the impacts to their community, society, and the world.

The *AP Computer Science Principles Curriculum Framework* specifies the course curriculum — the concepts and computational thinking practices central to the discipline of computer science — and is organized around the investigation of seven big ideas, all of which are fundamental principles essential to thrive in future college courses and a variety of computing and STEM (science, technology, engineering, mathematics) careers. Emphasizing these key big ideas helps students build a solid understanding and facility with computing and computational thinking. These integral understandings can be applied in further studies of computer science and provide a pathway for becoming a well-educated and informed citizen who understands how computer science impacts people and society.

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 in the following text.

▶ 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 computational thinking 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 computational thinking practices require students to learn to collaborate to build computational artifacts and communicate their purpose. Because the AP Computer Science Principles content and the computational thinking practices are equally important, each learning objective directly correlates to a computational thinking practice. This correlation to a computational thinking practice 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 major areas of study in the course are organized around seven **big ideas**, which encompass ideas foundational to studying computer science. 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 (LO)** 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 7.2.1 means it is from Big Idea 7, Enduring Understanding 7.2, and it is the first learning objective in this section). **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 essential knowledge (EK) statements are listed numerically in the column next to the correlated learning objective, and each one includes one or more 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 could 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.

Relationship between the Curriculum Framework and Assessment

The learning objectives (including the essential knowledge statements and computational thinking practices) will be the targets of assessment for the AP Computer Science Principles course. This assessment comprises two parts: the end-of-course AP Exam and the through-course AP assessment.

The AP Computer Science Principles Exam will be a multiple-choice, paper and pencil exam in which students will demonstrate achievement of the course learning objectives.

The through-course assessment comprises two AP Computer Science Principles performance tasks, which require students to explore the impacts of computing and create computational artifacts through programming. Like the AP Exam, the performance tasks are designed to gather evidence of student learning with regard to the learning objectives. Performance tasks assess student achievement in more robust ways than are available on a timed exam. Additionally, there are learning objectives that are more effectively measured in an authentic, "real-world" performance task.

For more information about the AP Computer Science Principles performance tasks, go to http://www.collegeboard.com/html/computerscience/index.html?MTG77-ED-1-apcs

On both the AP Computer Science Principles Exam and the through-course assessment (performance tasks), students will be asked to apply their understanding of the course learning objectives, including the essential knowledge statements and computational thinking practices.

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.

▸ 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 Web sites, 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.

▸ 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.

▸ 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.

▸ Justify appropriateness and correctness of a solution, model, or artifact.

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.

▸ 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 an 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.

▸ 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	Essential Knowledge
EU 1.1 Creative development can be an essential process for creating computational artifacts.	**LO 1.1.1** Apply a creative development process when creating computational artifacts. [P2]	**EK 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.
		EK 1.1.1B Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.
EU 1.2 Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem.	**LO 1.2.1** Create a computational artifact for creative expression. [P2]	**EK 1.2.1A** A computational artifact is something 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.
		EK 1.2.1B Creating computational artifacts requires understanding of and use of software tools and services.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 1.2 Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem. *(continued)*		**EK 1.2.1C** Computing tools and techniques are used to create computational artifacts and can include, but are not limited to, programming integrated development environments (IDEs), spreadsheets, 3D printers, or text editors.
		EK 1.2.1D A creatively developed computational artifact can be created by using nontraditional, nonprescribed computing techniques.
		EK 1.2.1E Creative expressions in a computational artifact can reflect personal expressions of ideas or interests.
	LO 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. [P2]	**EK 1.2.2A** Computing tools and techniques can enhance the process of finding a solution to a problem.
		EK 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.
	LO 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. [P2]	**EK 1.2.3A** Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.
		EK 1.2.3B Computation facilitates the creation and modification of computational artifacts with enhanced detail and precision.
		EK 1.2.3C Combining or modifying existing artifacts can show personal expression of ideas.
	LO 1.2.4 Collaborate in the creation of computational artifacts. [P6]	**EK 1.2.4A** A collaboratively created computational artifact reflects effort by more than one person.
		EK 1.2.4B Effective collaborative teams consider the use of online collaborative tools.
		EK 1.2.4C Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation.
		EK 1.2.4D Effective collaboration strategies enhance performance.
		EK 1.2.4E Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts.
		EK 1.2.4F A collaboratively created computational artifact can reflect personal expressions of ideas.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 1.2 Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem. *(continued)*	**LO 1.2.5** Analyze the correctness, usability, functionality, and suitability of computational artifacts. [P4]	**EK 1.2.5A** The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact.
		EK 1.2.5B A computational artifact may have weaknesses, mistakes, or errors depending on the type of artifact.
		EK 1.2.5C The functionality of a computational artifact may be related to how it is used or perceived.
		EK 1.2.5D The suitability (or appropriateness) of a computational artifact may be related to how it is used or perceived.
EU 1.3 Computing can extend traditional forms of human expression and experience.	**LO 1.3.1** Use computing tools and techniques for creative expression. [P2]	**EK 1.3.1A** Creating digital effects, images, audio, video, and animations has transformed industries.
		EK 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.
		EK 1.3.1C Digital images can be created by generating pixel patterns, manipulating existing digital images, or combining images.
		EK 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.
		EK 1.3.1E Computing enables creative exploration of both real and virtual phenomena.

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 requires students to use abstractions to model the world and communicate 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 changes, 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	Essential Knowledge
EU 2.1 A variety of abstractions built on binary sequences can be used to represent all digital data.	**LO 2.1.1** Describe the variety of abstractions used to represent data. [P3]	**EK 2.1.1A** Digital data is represented by abstractions at different levels.
		EK 2.1.1B At the lowest level, all digital data are represented by bits.
		EK 2.1.1C At a higher level, bits are grouped to represent abstractions, including but not limited to numbers, characters, and color.
		EK 2.1.1D Number bases, including binary, decimal, and hexadecimal, are used to represent and investigate digital data.
		EK 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.
		EXCLUSION STATEMENT (for EK 2.1.1E): Two's complement conversions are beyond the scope of this course and the AP Exam.
		EK 2.1.1F Hexadecimal (base 16) is used to represent digital data because hexadecimal representation uses fewer digits than binary.
		EK 2.1.1G Numbers can be converted from any base to any other base.
	LO 2.1.2 Explain how binary sequences are used to represent digital data. [P5]	**EK 2.1.2A** A finite representation is used to model the infinite mathematical concept of a number.
		EXCLUSION STATEMENT (for EK 2.1.2A): Binary representations of scientific notation are beyond the scope of this course and the AP Exam.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 2.1 A variety of abstractions built on binary sequences can be used to represent all digital data. *(continued)*		**EK 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. **EXCLUSION STATEMENT (for EK 2.1.2B):** Range limitations of any one language, compiler, or architecture are beyond the scope of this course and the AP Exam. **EK 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. **EK 2.1.2D** The interpretation of a binary sequence depends on how it is used. **EK 2.1.2E** A sequence of bits may represent instructions or data. **EK 2.1.2F** A sequence of bits may represent different types of data in different contexts.
EU 2.2 Multiple levels of abstraction are used to write programs or create other computational artifacts.	**LO 2.2.1** Develop an abstraction when writing a program or creating other computational artifacts. [P2]	**EK 2.2.1A** The process of developing an abstraction involves removing detail and generalizing functionality. **EK 2.2.1B** An abstraction extracts common features from specific examples in order to generalize concepts. **EK 2.2.1C** An abstraction generalizes functionality with input parameters that allow software reuse. **EXCLUSION STATEMENT (for EK 2.2.1C):** An understanding of the difference between value and reference parameters is beyond the scope of this course and the AP Exam.
	LO 2.2.2 Use multiple levels of abstraction to write programs. [P3]	**EK 2.2.2A** Software is developed using multiple levels of abstractions, such as constants, expressions, statements, procedures, and libraries. **EK 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.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 2.2 Multiple levels of abstraction are used to write programs or create other computational artifacts. *(continued)*	**LO 2.2.3** Identify multiple levels of abstractions that are used when writing programs. [P3]	**EK 2.2.3A** Different programming languages offer different levels of abstraction. **EXCLUSION STATEMENT (for EK 2.2.3A):** Knowledge of the abstraction capabilities of all programming languages is beyond the scope of this course and the AP Exam.
		EK 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.
		EK 2.2.3C Code in a programming language is often translated into code in another (lower level) language to be executed on a computer.
		EK 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.
		EK 2.2.3E Binary data is processed by physical layers of computing hardware, including gates, chips, and components.
		EK 2.2.3F A logic gate is a hardware abstraction that is modeled by a Boolean function. **EXCLUSION STATEMENT (for EK 2.2.3F):** Memorization of specific gate visual representations is beyond the scope of this course and the AP Exam.
		EK 2.2.3G A chip is an abstraction composed of low-level components and circuits that perform a specific function.
		EK 2.2.3H A hardware component can be low level like a transistor or high level like a video card.
		EK 2.2.3I Hardware is built using multiple levels of abstractions, such as transistors, logic gates, chips, memory, motherboards, special purpose cards, and storage devices.
		EK 2.2.3J Applications and systems are designed, developed, and analyzed using levels of hardware, software, and conceptual abstractions.
		EK 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.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 2.3 Models and simulations use abstraction to generate new understanding and knowledge.	**LO 2.3.1** Use models and simulations to represent phenomena. [P3]	**EK 2.3.1A** Models and simulations are simplified representations of more complex objects or phenomena.
		EK 2.3.1B Models may use different abstractions or levels of abstraction depending on the objects or phenomena being posed.
		EK 2.3.1C Models often omit unnecessary features of the objects or phenomena that are being modeled.
		EK 2.3.1D Simulations mimic real-world events without the cost or danger of building and testing the phenomena in the real world.
	LO 2.3.2 Use models and simulations to formulate, refine, and test hypotheses. [P3]	**EK 2.3.2A** Models and simulations facilitate the formulation and refinement of hypotheses related to the objects or phenomena under consideration.
		EK 2.3.2B Hypotheses are formulated to explain the objects or phenomena being modeled.
		EK 2.3.2C Hypotheses are refined by examining the insights that models and simulations provide into the objects or phenomena.
		EK 2.3.2D The results of simulations may generate new knowledge and new hypotheses related to the phenomena being modeled.
		EK 2.3.2E Simulations allow hypotheses to be tested without the constraints of the real world.
		EK 2.3.2F Simulations can facilitate extensive and rapid testing of models.
		EK 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.
		EK 2.3.2H Rapid and extensive testing allows models to be changed to accurately reflect the objects or phenomena being modeled.

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 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	Essential Knowledge
EU 3.1 People use computer programs to process information to gain insight and knowledge.	**LO 3.1.1** Use computers to process information, find patterns, and test hypotheses about digitally processed information to gain insight and knowledge. [P4]	**EK 3.1.1A** Computers are used in an iterative and interactive way when processing digital information to gain insight and knowledge.
		EK 3.1.1B Digital information can be filtered and cleaned by using computers to process information.
		EK 3.1.1C Combining data sources, clustering data, and data classification are part of the process of using computers to process information.
		EK 3.1.1D Insight and knowledge can be obtained from translating and transforming digitally represented information.
		EK 3.1.1E Patterns can emerge when data is transformed using computational tools.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 3.1 People use computer programs to process information to gain insight and knowledge. *(continued)*	**LO 3.1.2** Collaborate when processing information to gain insight and knowledge. [P6]	**EK 3.1.2A** Collaboration is an important part of solving data-driven problems.
		EK 3.1.2B Collaboration facilitates solving computational problems by applying multiple perspectives, experiences, and skill sets.
		EK 3.1.2C Communication between participants working on data-driven problems gives rise to enhanced insights and knowledge.
		EK 3.1.2D Collaboration in developing hypotheses and questions, and in testing hypotheses and answering questions, about data helps participants gain insight and knowledge.
		EK 3.1.2E Collaborating face-to-face and using online collaborative tools can facilitate processing information to gain insight and knowledge.
		EK 3.1.2F Investigating large data sets collaboratively can lead to insight and knowledge not obtained when working alone.
	LO 3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notations, and precise language. [P5]	**EK 3.1.3A** Visualization tools and software can communicate information about data.
		EK 3.1.3B Tables, diagrams, and textual displays can be used in communicating insight and knowledge gained from data.
		EK 3.1.3C Summaries of data analyzed computationally can be effective in communicating insight and knowledge gained from digitally represented information.
		EK 3.1.3D Transforming information can be effective in communicating knowledge gained from data.
		EK 3.1.3E Interactivity with data is an aspect of communicating.
EU 3.2 Computing facilitates exploration and the discovery of connections in information.	**LO 3.2.1** Extract information from data to discover and explain connections, patterns, or trends. [P1]	**EK 3.2.1A** Large data sets provide opportunities and challenges for extracting information and knowledge.
		EK 3.2.1B Large data sets provide opportunities for identifying trends, making connections in data, and solving problems.
		EK 3.2.1C Computing tools facilitate the discovery of connections in information within large data sets.
		EK 3.2.1D Search tools are essential for efficiently finding information.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 3.2 Computing facilitates exploration and the discovery of connections in information. *(continued)*		**EK 3.2.1E** Information filtering systems are important tools for finding information and recognizing patterns in the information.
		EK 3.2.1F Software tools, including spreadsheets and databases, help to efficiently organize and find trends in information.
		EXCLUSION STATEMENT (for EK 3.2.1F): Students are not expected to know specific formulas or options available in spreadsheet or database software packages.
		EK 3.2.1G Metadata is data about data.
		EK 3.2.1H Metadata can be descriptive data about an image, a Web page, or other complex objects.
		EK 3.2.1I Metadata can increase the effective use of data or data sets by providing additional information about various aspects of that data.
	LO 3.2.2. Use large data sets to explore and discover information and knowledge. [P3]	**EK 3.2.2A** Large data sets include data such as transactions, measurements, texts, sounds, images, and videos.
		EK 3.2.2B The storing, processing, and curating of large data sets is challenging.
		EK 3.2.2C Structuring large data sets for analysis can be challenging.
		EK 3.2.2D Maintaining privacy of large data sets containing personal information can be challenging.
		EK 3.2.2E Scalability of systems is an important consideration when data sets are large.
		EK 3.2.2F The size or scale of a system that stores data affects how that data set is used.
		EK 3.2.2G The effective use of large data sets requires computational solutions.
		EK 3.2.2H Analytical techniques to store, manage, transmit, and process data sets change as the size of data sets scale.
EU 3.3 There are trade-offs when representing information as digital data.	**LO 3.3.1** Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information. [P4]	**EK 3.3.1A** Digital data representations involve trade-offs related to storage, security, and privacy concerns.
		EK 3.3.1B Security concerns engender trade-offs in storing and transmitting information.
		EK 3.3.1C There are trade-offs in using lossy and lossless compression techniques for storing and transmitting data.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 3.3 There are trade-offs when representing information as digital data. *(continued)*		**EK 3.3.1D** Lossless data compression reduces the number of bits stored or transmitted but allows complete reconstruction of the original data.
		EK 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.
		EK 3.3.1F Security and privacy concerns arise with data containing personal information.
		EK 3.3.1G Data is stored in many formats depending on its characteristics (e.g., size and intended use).
		EK 3.3.1H The choice of storage media affects both the methods and costs of manipulating the data it contains.
		EK 3.3.1I Reading data and updating data have different storage requirements.

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 (Students will be able to …)	Essential Knowledge (Students will know that …)
EU 4.1 Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages.	**LO 4.1.1** Develop an algorithm for implementation in a program. [P2]	**EK 4.1.1A** Sequencing, selection, and iteration are building blocks of algorithms.
		EK 4.1.1B Sequencing is the application of each step of an algorithm in the order in which the statements are given.
		EK 4.1.1C Selection uses a Boolean condition to determine which of two parts of an algorithm is used.
		EK 4.1.1D Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
		EK 4.1.1E Algorithms can be combined to make new algorithms.
		EK 4.1.1F Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.
		EK 4.1.1G Knowledge of standard algorithms can help in constructing new algorithms.
		EK 4.1.1H Different algorithms can be developed to solve the same problem.
		EK 4.1.1I Developing a new algorithm to solve a problem can yield insight into the problem.
	LO 4.1.2 Express an algorithm in a language. [P5]	**EK 4.1.2A** Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
		EK 4.1.2B Natural language and pseudocode describe algorithms so that humans can understand them.
		EK 4.1.2C Algorithms described in programming languages can be executed on a computer.
		EK 4.1.2D Different languages are better suited for expressing different algorithms.
		EK 4.1.2E Some programming languages are designed for specific domains and are better for expressing algorithms in those domains.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 4.1 Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages. *(continued)*		**EK 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.
		EK 4.1.2G Every algorithm can be constructed using only sequencing, selection, and iteration.
		EK 4.1.2H Nearly all programming languages are equivalent in terms of being able to express any algorithm.
		EK 4.1.2I Clarity and readability are important considerations when expressing an algorithm in a language.
EU 4.2 Algorithms can solve many, but not all, computational problems.	**LO 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] **EXCLUSION STATEMENT (for LO 4.2.1):** Any discussion of nondeterministic polynomial (NP) is beyond the scope of this course and the AP Exam.	**EK 4.2.1A** Many problems can be solved in a reasonable time.
		EK 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.
		EK 4.2.1C Some problems cannot be solved in a reasonable time, even for small input sizes.
		EK 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.
	LO 4.2.2 Explain the difference between solvable and unsolvable problems in computer science. [P1] **EXCLUSION STATEMENT (for LO 4.2.2):** Determining whether a given problem is solvable or unsolvable is beyond the scope of this course and the AP Exam.	**EK 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.
		EK 4.2.2B Heuristics may be helpful for finding an approximate solution more quickly when exact methods are too slow.
		EXCLUSION STATEMENT (for EK 4.2.2B): Specific heuristic solutions are beyond the scope of this course and the AP Exam.
		EK 4.2.2C Some optimization problems such as "find the best" or "find the smallest" cannot be solved in a reasonable time but approximations to the optimal solution can.
		EK 4.2.2D Some problems cannot be solved using any algorithm.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 4.2 Algorithms can solve many, but not all, computational problems. *(continued)*	**LO 4.2.3** Explain the existence of undecidable problems in computer science. [P1]	**EK 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.
		EK 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?").
		EK 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.
		EXCLUSION STATEMENT (for EK 4.2.3C): Determining whether a given problem is undecidable is beyond the scope of this course and the AP Exam.
	LO 4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity. [P4]	**EK 4.2.4A** Determining an algorithm's efficiency is done by reasoning formally or mathematically about the algorithm.
		EK 4.2.4B Empirical analysis of an algorithm is done by implementing the algorithm and running it on different inputs.
		EK 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.
		EXCLUSION STATEMENT (for EK 4.2.4C): Formally proving program correctness is beyond the scope of this course and the AP Exam.
		EK 4.2.4D Different correct algorithms for the same problem can have different efficiencies.
		EK 4.2.4E Sometimes, more efficient algorithms are more complex.
		EK 4.2.4F Finding an efficient algorithm for a problem can help solve larger instances of the problem.
		EK 4.2.4G Efficiency includes both execution time and memory usage.
		EXCLUSION STATEMENT (for EK 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.
		EK 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.

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 particular programming language is selected based on appropriateness for a specific project or problem. The course acquaints students with fundamental concepts of programming that can be applied across a variety of projects and languages. As students learn language specifics for a given programming language, they 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	Essential Knowledge
EU 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).	**LO 5.1.1** Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge. [P2]	**EK 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.
		EK 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.
		EK 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.
		EK 5.1.1D Additional desired outcomes may be realized independently of the original purpose of the program.
		EK 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.
		EK 5.1.1F Advances in computing have generated and increased creativity in other fields.

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

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 5.2 People write programs to execute algorithms.	**LO 5.2.1** Explain how programs implement algorithms. [P3]	**EK 5.2.1A** Algorithms are implemented using program instructions that are processed during program execution.
		EK 5.2.1B Program instructions are executed sequentially.
		EK 5.2.1C Program instructions may involve variables that are initialized and updated, read, and written.
		EK 5.2.1D An understanding of instruction processing and program execution is useful for programming.
		EK 5.2.1E Program execution automates processes.
		EK 5.2.1F Processes use memory, a central processing unit (CPU), and input and output.
		EK 5.2.1G A process may execute by itself or with other processes.
		EK 5.2.1H A process may execute on one or several CPUs.
		EK 5.2.1I Executable programs increase the scale of problems that can be addressed.
		EK 5.2.1J Simple algorithms can solve a large set of problems when automated.
		EK 5.2.1K Improvements in algorithms, hardware, and software increase the kinds of problems and the size of problems solvable by programming.
EU 5.3 Programming is facilitated by appropriate abstractions.	**LO 5.3.1** Use abstraction to manage complexity in programs. [P3]	**EK 5.3.1A** Procedures are reusable programming abstractions.
		EK 5.3.1B A procedure is a named grouping of programming instructions.
		EK 5.3.1C Procedures reduce the complexity of writing and maintaining programs.
		EK 5.3.1D Procedures have names and may have parameters and return values.
		EK 5.3.1E Parameterization can generalize a specific solution.
		EK 5.3.1F Parameters generalize a solution by allowing a procedure to be used instead of duplicated code.
		EK 5.3.1G Parameters provide different values as input to procedures when they are called in a program.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 5.3 Programming is facilitated by appropriate abstractions. *(continued)*		**EK 5.3.1H** Data abstraction provides a means of separating behavior from implementation.
		EK 5.3.1I Strings and string operations, including concatenation and some form of substring, are common in many programs.
		EK 5.3.1J Integers and floating-point numbers are used in programs without requiring understanding of how they are implemented.
		EK 5.3.1K Lists and list operations, such as add, remove, and search, are common in many programs.
		EK 5.3.1L Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.
		EK 5.3.1M Application program interfaces (APIs) and libraries simplify complex programming tasks.
		EK 5.3.1N Documentation for an API/library is an important aspect of programming.
		EK 5.3.1O APIs connect software components, allowing them to communicate.
EU 5.4 Programs are developed, maintained, and used by people for different purposes.	**LO 5.4.1** Evaluate the correctness of a program. [P4]	**EK 5.4.1A** Program style can affect the determination of program correctness.
		EK 5.4.1B Duplicated code can make it harder to reason about a program.
		EK 5.4.1C Meaningful names for variables and procedures help people better understand programs.
		EK 5.4.1D Longer code blocks are harder to reason about than shorter code blocks in a program.
		EK 5.4.1E Locating and correcting errors in a program is called debugging the program.
		EK 5.4.1F Knowledge of what a program is supposed to do is required in order to find most program errors.
		EK 5.4.1G Examples of intended behavior on specific inputs help people understand what a program is supposed to do.
		EK 5.4.1H Visual displays (or different modalities) of program state can help in finding errors.
		EK 5.4.1I Programmers justify and explain a program's correctness.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 5.4 Programs are developed, maintained, and used by people for different purposes. *(continued)*		EK 5.4.1J Justification can include a written explanation about how a program meets its specifications.
		EK 5.4.1K Correctness of a program depends on correctness of program components, including code blocks and procedures.
		EK 5.4.1L An explanation of a program helps people understand the functionality and purpose of it.
		EK 5.4.1M The functionality of a program is often described by how a user interacts with it.
		EK 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.
EU 5.5 Programming uses mathematical and logical concepts.	LO 5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1]	EK 5.5.1A Numbers and numerical concepts are fundamental to programming.
		EK 5.5.1B Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.
		EXCLUSION STATEMENT (for EK 5.5.1B): Specific range limitations of all programming languages are beyond the scope of this course and the AP Exam.
		EK 5.5.1C Real numbers are approximated by floating-point representations that do not necessarily have infinite precision.
		EXCLUSION STATEMENT (for EK 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.
		EK 5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.
		EK 5.5.1E Logical concepts and Boolean algebra are fundamental to programming.
		EK 5.5.1F Compound expressions using *and*, *or*, and *not* are part of most programming languages.
		EK 5.5.1G Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.
		EK 5.5.1H Computational methods may use lists and collections to solve problems.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 5.5 Programming uses mathematical and logical concepts. *(continued)*		EK 5.5.1I Lists and other collections can be treated as abstract data types (ADTs) in developing programs.
		EK 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.

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 on 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 (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
EU 6.1 The Internet is a network of autonomous systems.	LO 6.1.1 Explain the abstractions in the Internet and how the Internet functions. [P3] **EXCLUSION STATEMENT (for 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.	EK 6.1.1A The Internet connects devices and networks all over the world.
		EK 6.1.1B An end-to-end architecture facilitates connecting new devices and networks on the Internet.
		EK 6.1.1C Devices and networks that make up the Internet are connected and communicate using addresses and protocols.
		EK 6.1.1D The Internet and the systems built on it facilitate collaboration.
		EK 6.1.1E Connecting new devices to the Internet is enabled by assignment of an Internet protocol (IP) address.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 6.1 The Internet is a network of autonomous systems. *(continued)*		**EK 6.1.1F** The Internet is built on evolving standards, including those for addresses and names.
		EXCLUSION STATEMENT (for EK 6.1.1F): Specific details of any particular standard for addresses are beyond the scope of this course and the AP Exam.
		EK 6.1.1G The domain name system (DNS) translates names to IP addresses.
		EK 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.
		EK 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).
EU 6.2 Characteristics of the Internet influence the systems built on it.	**LO 6.2.1** Explain characteristics of the Internet and the systems built on it. [P5]	**EK 6.2.1A** The Internet and the systems built on it are hierarchical and redundant.
		EK 6.2.1B The domain name syntax is hierarchical.
		EK 6.2.1C IP addresses are hierarchical.
		EK 6.2.1D Routing on the Internet is fault tolerant and redundant.
	LO 6.2.2 Explain how the characteristics of the Internet influence the systems built on it. [P4]	**EK 6.2.2A** Hierarchy and redundancy help systems scale.
		EK 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.
		EK 6.2.2C Hierarchy in the DNS helps that system scale.
		EK 6.2.2D Interfaces and protocols enable widespread use of the Internet.
		EK 6.2.2E Open standards fuel the growth of the Internet.
		EK 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.
		EXCLUSION STATEMENT (for EK 6.2.2F): Specific details of any particular packet-switching system are beyond the scope of this course and the AP Exam.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 6.2 Characteristics of the Internet influence the systems built on it. *(continued)*		**EK 6.2.2G** Standards for packets and routing include transmission control protocol/Internet protocol (TCP/IP). **EXCLUSION STATEMENT (for EK 6.2.2G):** Specific technical details of how TCP/IP works are beyond the scope of this course and the AP Exam.
		EK 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). **EXCLUSION STATEMENT (for EK 6.2.2H):** Understanding the technical aspects of how SSL/TLS works is beyond the scope of this course and the AP Exam.
		EK 6.2.2I The size and speed of systems affect their use.
		EK 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.
		EK 6.2.2K The latency of a system is the time elapsed between the transmission and the receipt of a request.
EU 6.3 Cybersecurity is an important concern for the Internet and the systems built on it.	**LO 6.3.1** Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it. [P1]	**EK 6.3.1A** The trust model of the Internet involves trade-offs.
		EK 6.3.1B The DNS was not designed to be completely secure.
		EK 6.3.1C Implementing cybersecurity has software, hardware, and human components.
		EK 6.3.1D Cyber warfare and cyber crime have widespread and potentially devastating effects.
		EK 6.3.1E Distributed denial-of-service attacks (DDoS) compromise a target by flooding it with requests from multiple systems.
		EK 6.3.1F Phishing, viruses, and other attacks have human and software components.
		EK 6.3.1G Antivirus software and firewalls can help prevent unauthorized access to private data.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 6.3 Cybersecurity is an important concern for the Internet and the systems built on it. *(continued)*		EK 6.3.1H Cryptography is essential to many models of cybersecurity.
		EK 6.3.1I Cryptography has a mathematical foundation.
		EXCLUSION STATEMENT (for EK 6.3.1I): Specific mathematical functions used in cryptography are beyond the scope of this course and the AP Exam.
		EK 6.3.1J Open standards help ensure cryptography is secure.
		EK 6.3.1K Symmetric encryption is a method of encryption involving one key for encryption and decryption.
		EXCLUSION STATEMENT (for EK 6.3.1K): The methods used in encryption are beyond the scope of this course and the AP Exam.
		EK 6.3.1L Public key encryption, which is not symmetric, is an encryption method that is widely used because of the functionality it provides.
		EXCLUSION STATEMENT (for EK 6.3.1L): The mathematical methods used in public key cryptography are beyond the scope of this course and the AP Exam.
		EK 6.3.1M Certificate authorities (CAs) issue digital certificates that validate the ownership of encrypted keys used in secured communications and are based on a trust model.
		EXCLUSION STATEMENT (for EK 6.3.1M): The technical details of the process CAs follow are beyond the scope of this course and the AP Exam.

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?

Enduring Understandings (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
EU 7.1 Computing enhances communication, interaction, and cognition.	**LO 7.1.1** Explain how computing innovations affect communication, interaction, and cognition. [P4]	**EK 7.1.1A** Email, SMS, and chat have fostered new ways to communicate and collaborate.
		EK 7.1.1B Video conferencing and video chat have fostered new ways to communicate and collaborate.
		EK 7.1.1C Social media continues to evolve and fosters new ways to communicate.
		EXCLUSION STATEMENT (for EK 7.1.1C): Detailed knowledge of any social media site is beyond the scope of this course and the AP Exam.
		EK 7.1.1D Cloud computing fosters new ways to communicate and collaborate.
		EK 7.1.1E Widespread access to information facilitates the identification of problems, development of solutions, and dissemination of results.
		EK 7.1.1F Public data provides widespread access and enables solutions to identified problems.
		EK 7.1.1G Search trends are predictors.
		EK 7.1.1H Social media, such as blogs and Twitter, have enhanced dissemination.
		EK 7.1.1I Global Positioning System (GPS) and related technologies have changed how humans travel, navigate, and find information related to geolocation.
		EK 7.1.1J Sensor networks facilitate new ways of interacting with the environment and with physical systems.
		EK 7.1.1K Smart grids, smart buildings, and smart transportation are changing and facilitating human capabilities.
		EK 7.1.1L Computing contributes to many assistive technologies that enhance human capabilities.
		EK 7.1.1M The Internet and the Web have enhanced methods of and opportunities for communication and collaboration.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 7.1 Computing enhances communication, interaction, and cognition. *(continued)*		**EK 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.
		EK 7.1.1O The Internet and the Web have impacted productivity, positively and negatively, in many areas.
	LO 7.1.2 Explain how people participate in a problem-solving process that scales. [P4]	**EK 7.1.2A** Distributed solutions must scale to solve some problems.
		EK 7.1.2B Science has been impacted by using scale and "citizen science" to solve scientific problems using home computers in scientific research.
		EK 7.1.2C Human computation harnesses contributions from many humans to solve problems related to digital data and the Web.
		EK 7.1.2D Human capabilities are enhanced by digitally enabled collaboration.
		EK 7.1.2E Some online services use the contributions of many people to benefit both individuals and society.
		EK 7.1.2F Crowdsourcing offers new models for collaboration, such as connecting people with jobs and businesses with funding.
		EK 7.1.2G The move from desktop computers to a proliferation of always-on mobile computers is leading to new applications.
EU 7.2 Computing enables innovation in nearly every field.	**LO 7.2.1** Explain how computing has impacted innovations in other fields. [P1]	**EK 7.2.1A** Machine learning and data mining have enabled innovation in medicine, business, and science.
		EK 7.2.1B Scientific computing has enabled innovation in science and business.
		EK 7.2.1C Computing enables innovation by providing the ability to access and share information.
		EK 7.2.1D Open access and Creative Commons have enabled broad access to digital information.
		EK 7.2.1E Open and curated scientific databases have benefited scientific researchers.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 7.2 Computing enables innovation in nearly every field. *(continued)*		**EK 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.
		EK 7.2.1G Advances in computing as an enabling technology have generated and increased the creativity in other fields.
EU 7.3 Computing has a global affect — both beneficial and harmful — on people and society.	**LO 7.3.1** Analyze the beneficial and harmful effects of computing. [P4]	**EK 7.3.1A** Innovations enabled by computing raise legal and ethical concerns.
		EK 7.3.1B Commercial access to music and movie downloads and streaming raises legal and ethical concerns.
		EK 7.3.1C Access to digital content via peer-to-peer networks raises legal and ethical concerns.
		EK 7.3.1D Both authenticated and anonymous access to digital information raise legal and ethical concerns.
		EK 7.3.1E Commercial and governmental censorship of digital information raise legal and ethical concerns.
		EK 7.3.1F Open source and licensing of software and content raise legal and ethical concerns.
		EK 7.3.1G Privacy and security concerns arise in the development and use of computational systems and artifacts.
		EK 7.3.1H Aggregation of information, such as geolocation, cookies, and browsing history, raises privacy and security concerns.
		EK 7.3.1I Anonymity in online interactions can be enabled through the use of online anonymity software and proxy servers.
		EK 7.3.1J Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions.
		EK 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.
		EK 7.3.1L Commercial and governmental curation of information may be exploited if privacy and other protections are ignored.
		EK 7.3.1M Targeted advertising is used to help individuals, but it can be misused at both individual and aggregate levels.

Enduring Understandings	Learning Objectives	Essential Knowledge
EU 7.3 Computing has a global affect — both beneficial and harmful — on people and society. *(continued)*		**EK 7.3.1N** Widespread access to digitized information raises questions about intellectual property.
		EK 7.3.1O Creation of digital audio, video, and textual content by combining existing content has been impacted by copyright concerns.
		EK 7.3.1P The Digital Millennium Copyright Act (DMCA) has been a benefit and a challenge in making copyrighted digital material widely available.
		EK 7.3.1Q Open source and free software have practical, business, and ethical impacts on widespread access to programs, libraries, and code.
EU 7.4 Computing innovations influence and are influenced by the economic, social, and cultural contexts in which they are designed and used.	**LO 7.4.1** Explain the connections between computing and economic, social, and cultural contexts. [P1]	**EK 7.4.1A** The innovation and impact of social media and online access varies in different countries and in different socioeconomic groups.
		EK 7.4.1B Mobile, wireless, and networked computing have an impact on innovation throughout the world.
		EK 7.4.1C The global distribution of computing resources raises issues of equity, access, and power.
		EK 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.
		EK 7.4.1E Networks and infrastructure are supported by both commercial and governmental initiatives.

The AP Computer Science Principles Assessment

The AP Computer Science Principles assessment consists of two parts: a through-course assessment and the end-of-course AP Exam. Both of these parts will measure student achievement of the course learning objectives. For the through-course assessment, students will upload digital artifacts and written responses via a Web-based digital application. The end-of-course AP Exam will be a paper and pencil exam.

Through-Course Assessment

The through-course assessment is a set of performance tasks designed to gather evidence of student proficiency in the learning objectives. Performance tasks assess student achievement in more "real-world" ways than are available on a timed exam. In addition, there are learning objectives that are more effectively measured in a performance task, such as those included in the Creativity big idea.

The performance tasks are summative assessments, and will be completed in the classroom.

The two performance tasks are:

Explore – Implications of Computing Innovations
Students explore the impacts of computing on social, economic, and cultural areas of our lives.

Create – Applications from Ideas
Students create computational artifacts through the design and development of programs.

Prior to administering the performance tasks, teachers should prepare their students by teaching the skills embodied in the learning objectives and the content articulated in the essential knowledge statements. Instruction may include practicing the performance tasks before administering them to students. Once a teacher administers a performance task with the intent to submit student artifacts for AP scoring purposes, students must complete the task without assistance from the teacher.

Distinguishing features of the performance tasks include the following:

▸ Each performance task covers numerous learning objectives, distributed across several big ideas.

▸ The *Create* performance task requires both collaborative and individual effort as well as reflections on each student's contribution to the task.

▸ Each task requires students to describe or analyze their work, whether the work includes research, the creation of an artifact (e.g., a video, spreadsheet, graph, or electronic slide show), or the creation of a program.

For the latest pilot (DRAFT) versions of the AP Computer Science Principles performance tasks and rubrics, go to **http://www.collegeboard.com/html /computerscience/index.html?excmpid=MTG77-ED-1-apcs**

AP Exam

The AP Computer Science Exam is 100 minutes long and includes approximately 74 multiple-choice questions, presented as either discrete questions or in sets. There are two types of multiple-choice questions:

▶ Single-Select Multiple-Choice: *Students select 1 answer from among 4 options.*

▶ Multiple-Select Multiple-Choice: *Students select 2 answers from among 4 options.*

Sample Exam Questions

To elicit evidence of student achievement of the course learning objectives, exam questions assess both the application of the computational thinking practices and an understanding of the big ideas. Exam questions may assess achievement of multiple learning objectives. They may also address content from more than one essential knowledge statement. Exam questions may be accompanied by non-textual stimulus material such as diagrams, charts, or other graphical illustrations.

The sample questions that follow illustrate the relationship between the curriculum framework and the AP Computer Science Principles Exam and serve as examples of the types of questions that will appear on the exam.

Each question is accompanied by a table containing the enduring understandings, learning objectives and essential knowledge statements that the question addresses. Note that in the cases where multiple learning objectives are provided for a question, the primary one is listed first.

1. Which of the following are examples of how digital audio tools have transformed the music recording industry?

   I. Digital audio tools have made it easier to record professional-quality audio on a home computer instead of in a recording studio.

   II. Digital audio tools have made it easier to create new sounds by sampling existing sounds.

   III. Digital audio tools have made it easier to synthesize new sounds without having to first record them with a microphone.

   (A) I only

   (B) I and II only

   (C) II and III only

   (D) I, II, and III

Answer: D

Enduring Understandings	Learning Objectives	Essential Knowledge
1.3 Computing can extend traditional forms of human expression and experience.	**1.3.1** Use computing tools and techniques for creative expression. [P2]	**1.3.1A** Creating digital effects, images, audio, video, and animations has transformed industries.
		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.
1.2 Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem.	**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.

2. The most common method for representing color with computing applications is to control the amounts of red, green, and blue that are mixed together to create a desired color. A color can be represented by an RGB (Red, Green, Blue) triplet, which consists of three numbers between 0 and 255. The first number in a triplet represents the amount of red, the second number represents the amount of green, and the third number represents the amount of blue. For example, the RGB triplet (88, 50, 250) represents a predominantly blue color because there is more blue than there is red or green.

In many cases, the values in an RGB triplet are converted to a binary or hexadecimal representation.

Which of the following RGB triplets represents a predominantly red color?

(A) The decimal RGB values (42, 200, 30)

(B) The decimal RGB values (250, 250, 250)

(C) The hexadecimal RGB values (18, CA, 1E)

(D) The binary RGB values (11101010, 01001000, 00011101)

Answer: D

Enduring Understandings	Learning Objectives	Essential Knowledge
2.1 A variety of abstractions built upon binary sequences can be used to represent all digital data.	**2.1.1** Describe the variety of abstractions used to represent data. [P3]	**2.1.1B** At the lowest level, all digital data are represented by bits.
		2.1.1C At a higher level, bits are grouped to represent abstractions, including but not limited to numbers, characters, and color.
		2.1.1D Number bases, including binary, decimal, and hexadecimal, are used to represent and investigate digital data.
		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.
		2.1.1F Hexadecimal (base 16) is used to represent digital data because hexadecimal representation uses fewer digits than binary.
		2.1.1G Numbers can be converted from any base to any other base.

3. The music director of a college radio station gathered data to analyze the songs that were played on the station during the last year. A song was listed in the data every time it was played. The following data was collected for each song:

- The name of the artist

- The song title

- The album title

- The genre

The music director wants to find the number of **unique** songs played in the jazz genre. Consider the two proposed algorithms below.

Algorithm I: Filter the data by creating a list of songs only in the jazz genre. Sort the list of jazz songs by song title. Set a counter to 0. Iterate through the sorted list. If a song is different from the previous song in the list (or if it is the first song in the list), increment the counter by 1.

Algorithm II: Create a new list of the data sorted by song title. Iterate through the sorted list. Every time a song is the same as the previous song in the list, delete the duplicate song from the list. Set a counter to 0. Iterate through the remaining list of songs. Each time jazz is listed as the genre, increment the counter by 1.

Which algorithm, if any, can the music director use to find the number of **unique** songs played in the jazz genre?

(A) I only

(B) II only

(C) Both I and II

(D) Neither I nor II

Answer: C

Enduring Understandings	Learning Objectives	Essential Knowledge
3.2 Computing facilitates exploration and the discovery of connections in information.	**3.2.1** Extract information from data to discover and explain connections, patterns, or trends. [P1]	**3.2.1B** Large data sets provide opportunities for identifying trends, making connections in data, and solving problems.
		3.2.1C Computing tools facilitate the discovery of connections in information within large data sets.
		3.2.1E Information filtering systems are important tools for finding information and recognizing patterns in the information.
		3.2.1F Software tools, including spreadsheets and databases, help to efficiently organize and find trends in information.
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]	**3.1.1B** Digital information can be filtered and cleaned by using computers to process information.
		3.1.1D Insight and knowledge can be obtained from translating and transforming digitally represented information.
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.
		4.1.1H Different algorithms can be developed to solve the same problem.

4. Suppose that every personal music playing device will be modified so that whenever a song is played, the device will anonymously upload the following metadata to a shared centralized database:

- The name of the song

- The artist who recorded the song

- The time of day that the song is played

Which of the following would NOT be possible if only the database is used?

(A) Music companies will be able to determine the artists who are rising in popularity.

(B) Artists will be able to determine which of their songs are most popular.

(C) Users will be able to compare their personal listening preferences to those of their friends.

(D) Users will be able to compare their personal listening preferences to the listening trends of the entire database.

Answer: C

Enduring Understandings	Learning Objectives	Essential Knowledge
3.2 Computing facilitates exploration and the discovery of connections in information.	**3.2.1** Extract information from data to discover and explain connections, patterns, or trends. [P1]	**3.2.1G** Metadata is data about data.
		3.2.1H Metadata can be descriptive data about an image, a Web page, or other complex objects.
		3.2.1I Metadata can increase the effective use of data or data sets by providing additional information about various aspects of that data.
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]	**7.3.1J** Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions.
		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.

5. Suppose that every personal music playing device will be modified so that whenever a song starts playing, the device will anonymously upload the following metadata to a shared centralized database:

   - The name of the song

   - The artist who recorded the song

   - The time of the upload

   - The location of the device during the upload

   Which of the following pieces of information will be possible to determine if only the information in the database is used?

   Select <u>two</u> answers.

   (A) A list of individuals who enjoy listening to a particular artist

   (B) The artists who are rising in popularity

   (C) The amount of money made by a certain artist

   (D) The city with the greatest number of uploads

Answer: B, D

Enduring Understandings	Learning Objectives	Essential Knowledge
3.2 Computing facilitates exploration and the discovery of connections in information.	**3.2.1** Extract information from data to discover and explain connections, patterns, or trends. [P1]	**3.2.1G** Metadata is data about data.
		3.2.1H Metadata can be descriptive data about an image, Web page, or other complex objects.
		3.2.1I Metadata can increase the effective use of data or data sets by providing additional information about various aspects of that data.
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]	**7.3.1J** Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions.
		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.

6. The question below uses a simple programming language, with the following instructions.

Instruction	Explanation
`random(a, b)`	Generates a random integer between `a` and `b`, inclusive.
`a + b`	Evaluates to the sum of the numbers `a` and `b`.
`a ← b`	Assigns the value of `b` to the variable `a`.
`REPEAT n TIMES { }`	The block of instructions contained between the braces `{ }` is repeated `n` times.
`display(expression)`	Displays the value of `expression`.

Consider the goal of simulating the results of flipping a fair coin 10 times, and displaying the number of times the coin lands on heads. Which of the following code segments can be used to accomplish the goal?

(A)
```
sum ← 0
REPEAT 10 TIMES
{
     sum ← sum + random(0, 1)
}
display(sum)
```

(B)
```
REPEAT 10 TIMES
{
     sum ← 0
     sum ← sum + random(0, 1)
}
display(sum)
```

(C)
```
sum ← 0
REPEAT 10 TIMES
{
     sum ← sum + random(1, 2)
}
display(sum)
```

(D)
```
REPEAT 10 TIMES
{
     sum ← 0
     sum ← sum + random(1, 2)
}
display(sum)
```

Answer: A

Enduring Understandings	Learning Objectives	Essential Knowledge
4.1 Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages.	**4.1.2** Express an algorithm in a language. [P5]	**4.1.2A** Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
		4.1.2B Natural language and pseudocode describe algorithms so that humans can understand them.
		4.1.2C Algorithms described in programming languages can be executed on a computer.
2.3 Models and simulations use abstraction to generate new understanding and knowledge.	**2.3.1** Use models and simulations to represent phenomena. [P3]	**2.3.1A** Models and simulations are simplified representations of more complex objects or phenomena.
		2.3.1B Models may use different abstractions or levels of abstraction depending on the objects or phenomena being posed.
5.2 People write programs to execute algorithms.	**5.2.1** Explain how programs implement algorithms. [P3]	**5.2.1B** Program instructions are executed sequentially.
		5.2.1C Program instructions may involve variables that are initialized and updated, read, and written.
5.5 Programming uses mathematical and logical concepts.	**5.5.1** Employ appropriate mathematical and logical concepts in programming. [P1]	**5.5.1A** Numbers and numerical concepts are fundamental to programming.
		5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.

7. The question below uses a simple programming language, with the following instructions.

Instruction	Explanation
`random(a, b)`	Generates a random integer between `a` and `b`, inclusive.
`a + b`	Evaluates to the sum of the numbers `a` and `b`.
`a ← b`	Assigns the value of `b` to the variable `a`.
`REPEAT n TIMES { }`	The block of instructions contained between the braces `{ }` is repeated `n` times.
`display(expression)`	Displays the value of `expression`.

Consider the goal of simulating the results of rolling a number cube (numbered 1 to 6) two times, and displaying the sum of the values obtained from the two rolls. Which of the following code segments will produce the appropriate results?

I. `display (random(1, 12))`

II. `display (random(1, 6) + random(1, 6))`

III.
```
sum ← 0
REPEAT 2 TIMES
{
      sum ← sum + random(1, 6)
}
display(sum)
```

(A) I only

(B) I and III only

(C) II and III only

(D) I, II, and III

Answer: C

Enduring Understandings	Learning Objectives	Essential Knowledge
4.1 Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages.	**4.1.2** Express an algorithm in a language. [P5]	**4.1.2A** Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.
		4.1.2B Natural language and pseudocode describe algorithms so that humans can understand them.
		4.1.2C Algorithms described in programming languages can be executed on a computer.
	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.
		4.1.1D Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.
		4.1.1H Different algorithms can be developed to solve the same problem.
2.3 Models and simulations use abstraction to generate new understanding and knowledge.	**2.3.1** Use models and simulations to represent phenomena. [P3]	**2.3.1A** Models and simulations are simplified representations of more complex objects or phenomena.
		2.3.1B Models may use different abstractions or levels of abstraction depending on the objects or phenomena being posed.
5.2 People write programs to execute algorithms.	**5.2.1** Explain how programs implement algorithms. [P3]	**5.2.1B** Program instructions are executed sequentially.
		5.2.1C Program instructions may involve variables that are initialized and updated, read, and written.
5.5 Programming uses mathematical and logical concepts.	**5.5.1** Employ appropriate mathematical and logical concepts in programming. [P1]	**5.5.1A** Numbers and numerical concepts are fundamental to programming.
		5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.

8. The question below uses a simple programming language with the following instructions.

Instruction	Explanation
`a + b`	Evaluates to the sum of the numbers `a` and `b`.
`a ← b`	Assigns the value of `b` to the variable `a`.
`isPositive(n)`	Evaluates to `true` if `n` is greater than 0; otherwise evaluates to `false`.
`FOR EACH n IN list { }`	The variable `n` is assigned the value of each list element sequentially. The block of instructions between the braces `{ }` is executed once for each assignment of `n`.
`IF (condition) instruction`	The `instruction` is executed once if `condition` is `true`; no action is taken if `condition` is `false`.
`display(expression)`	Displays the value of `expression`.

Consider the goal of counting the number of positive-valued integers in a list called numbers. Which of the following code segments can be used to accomplish the goal?

(A)
```
count ← 0
FOR EACH value IN numbers
{
        IF (isPositive (value)) count ← count + 1
}
display(count)
```

(B)
```
FOR EACH value IN numbers
{
        count ← 0
        IF (isPositive (value)) count ← count + 1
}
display(count)
```

(C)
```
count ← 0
FOR EACH value IN numbers
{
        IF (isPositive (count)) increment(value)
}
display(count)
```

(D)
```
FOR EACH value IN numbers
{
        count ← 0
        IF (isPositive (count)) increment(value)
}
display(count)
```

Answer: A

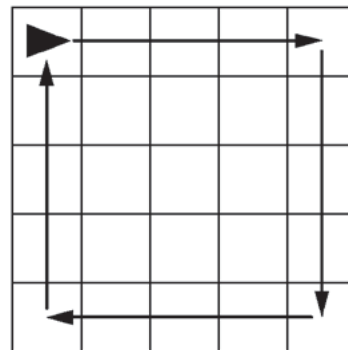| Enduring Understandings	Learning Objectives	Essential Knowledge
5.5 Programming uses mathematical and logical concepts.	**5.5.1** Employ appropriate mathematical and logical concepts in programming. [P1]	**5.5.1I** Lists and other collections can be treated as abstract data types (ADTs) in developing programs.
		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.
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.
5.3 Programming is facilitated by appropriate abstractions.	**5.3.1** Use abstraction to manage complexity in programs. [P3]	**5.3.1A** Procedures are reusable programming abstractions.
		5.3.1B A procedure is a named grouping of programming instructions.
		5.3.1G Parameters provide different values as input to procedures when they are called in a program.

9. The question below uses a robot in a grid of squares. The robot is represented as a triangle, which is initially facing toward the right side of the grid. The robot is moved according to the following instructions.

Instruction	Explanation
`Move Forward`	The robot moves one square forward in the direction it is facing.
`Rotate Right`	The robot rotates in place 90 degrees clockwise (i.e., makes an in-place right turn).
`Rotate Left`	The robot rotates in place 90 degrees counterclockwise (i.e., makes an in-place left turn).
`REPEAT n TIMES`	The block of instructions contained between the braces `{ }` is repeated `n` times.

Consider the following incorrect program, which is intended to move the robot around the perimeter of the grid below, as indicated by the arrows.

```
Line 1: REPEAT 4 TIMES
           {
Line 2:      Move Forward
Line 3:      Rotate Right
Line 4:      REPEAT 4 TIMES
             {
Line 5:          Move Forward
             }
Line 6:      Rotate Right
           }
```

Which lines of code should be removed so that the program will work as intended?
Select <u>two</u> answers.

(A) Line 2

(B) Line 3

(C) Line 4

(D) Line 5

Answer: A, B

Enduring Understandings	Learning Objectives	Essential Knowledge
5.4 Programs are developed, maintained, and used by people for different purposes.	**5.4.1** Evaluate the correctness of a program. [P4]	**5.4.1E** Locating and correcting errors in a program is called debugging the program.
		5.4.1F Knowledge of what a program is supposed to do is required in order to find most program errors.
		5.4.1G Examples of intended behavior on specific inputs help people understand what a program is supposed to do.
		5.4.1K Correctness of a program depends on correctness of program components, including code blocks and procedures.

10. Which of the following statements correctly explain how the Internet is able to facilitate communication at a large scale?

   I. A central monitoring computer is used to track and maintain the connections of the Internet.

   II. Data is routed between points in multiple ways so that if a connection fails, the data can be rerouted around the inoperative connections.

   III. Protocols for packets and routing are used so that computers from different manufacturers can communicate in a standard way.

   (A) I and II only

   (B) I and III only

   (C) II and III only

   (D) I, II, and III

Answer: C

Enduring Understandings	Learning Objectives	Essential Knowledge
6.2 Characteristics of the Internet influence the systems built on it.	**6.2.2** Explain how the characteristics of the Internet influence the systems built on it. [P4]	**6.2.2A** Hierarchy and redundancy help systems scale.
		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.
		6.2.2D Interfaces and protocols enable widespread use of the Internet.

apcentral.collegeboard.org