Course and Exam Description

AP®

AP® Computer Science Principles

Including the Curriculum Framework

Updated Fall 2017
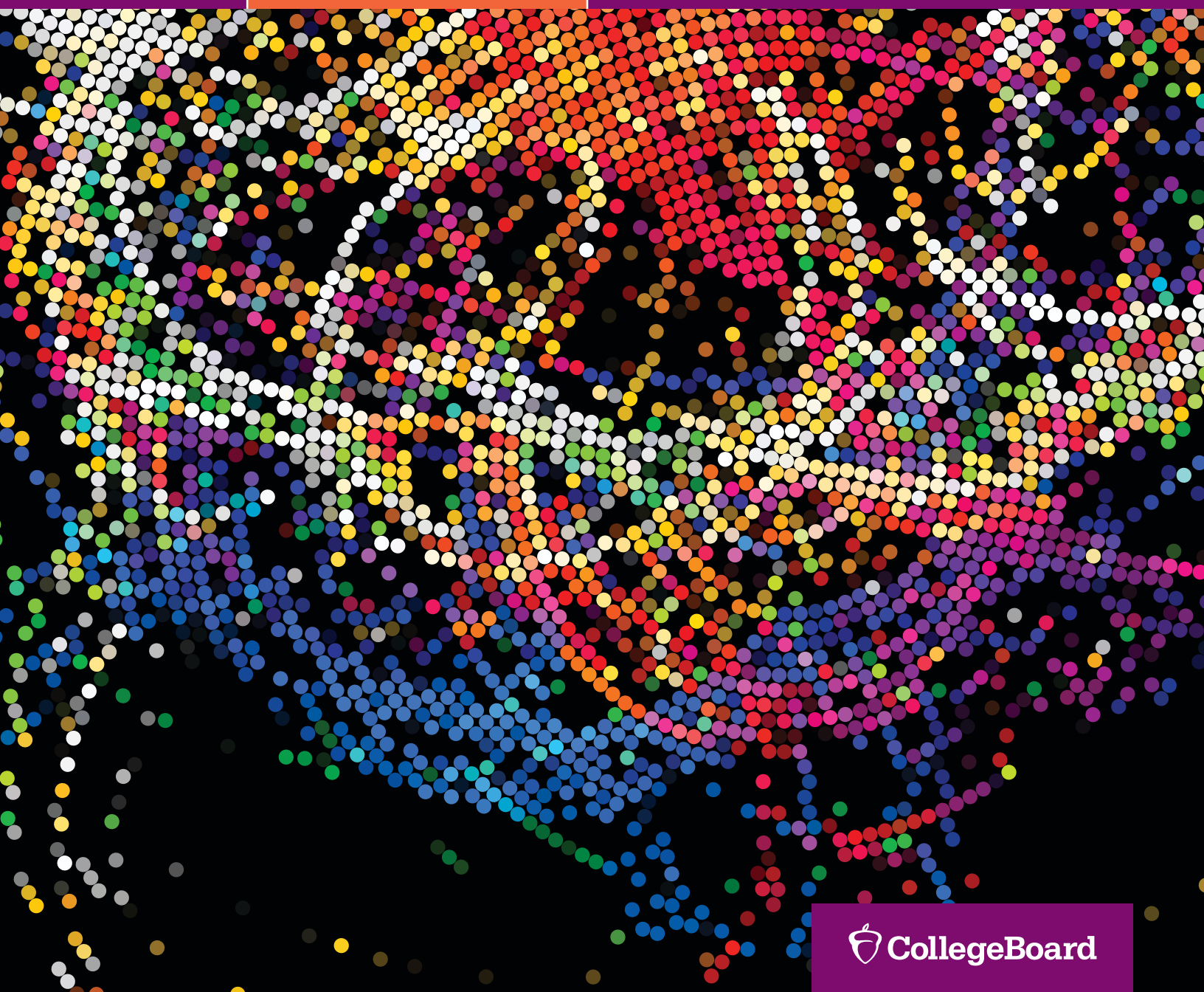
CollegeBoard

AP® Computer Science Principles

Course and Exam Description
Updated Fall 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 underrepresented. 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.

Contents

Preface

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. Students develop innovative computational artifacts using the same creative processes 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 emphasizes the vital impact advances in computing have on people and society. The course goes beyond the study of machines and systems and gives students the opportunity to investigate computing innovations that span a variety of interests and to examine the ethical implications of these new technologies.

In partnership with the National Science Foundation, the AP Program collaborated with secondary and postsecondary educators and members of computer science educational professional organizations to develop the AP Computer Science Principles curriculum framework.

This new AP Computer Science Principles course is complementary to AP Computer Science A. Students can take these courses in any order or at the same time, as schedules permit. Both courses include rigorous computer science content and skills that can be built on to complete further science, technology, engineering, mathematics, and computing studies. It is important to note that the AP Computer Science Principles course does not have a designated programming language. Teachers have the flexibility to choose a programming language(s) that is most appropriate for their students to use in the classroom.

Acknowledgments

Larry Snyder, *University of Washington*

Lynn Andrea Stein, *Olin College*

Chris Stephenson, *Computer Science Teachers Association*

Fran Trees, *Rutgers University**

Cameron Wilson, *Association for Computing Machinery*

AP Curriculum and Content Development

Lien Diaz, *Senior Director, AP Curriculum, Instruction, and Assessment*

AP Instructional Design and Professional Development

Crystal Furman, *Director, AP Curriculum, Instruction, and Assessment*

---

*Members of the original Development Committee are marked with an asterisk (*).

About AP®

The College Board's Advanced Placement Program® (AP®) enables students to pursue college-level studies while still in high school. Through more than 30 courses, each culminating in a rigorous exam, AP provides willing and academically prepared students with the opportunity to earn college credit and/or advanced placement. Taking AP courses also demonstrates to college admission officers that students have sought out the most rigorous course work available to them.

Each AP course is modeled upon a comparable college course, and college and university faculty play a vital role in ensuring that AP courses align with college-level standards. Talented and dedicated AP teachers help AP students in classrooms around the world develop and apply the content knowledge and skills they will need later in college.

Each AP course concludes with a college-level assessment developed and scored by college and university faculty, as well as experienced AP teachers. AP Exams are an essential part of the AP experience, enabling students to demonstrate their mastery of college-level course work. Most four-year colleges and universities in the United States and universities in more than 60 countries recognize AP in the admission process and grant students credit, placement, or both on the basis of successful AP Exam scores. Visit **www.collegeboard.org/apcreditpolicy** to view AP credit and placement policies at more than 1,000 colleges and universities.

Performing well on an AP Exam means more than just the successful completion of a course; it is a gateway to success in college. Research consistently shows that students who receive a score of 3 or higher on AP Exams typically experience greater academic success in college and have higher graduation rates than their non-AP peers.[1] Additional AP studies are available at **www.collegeboard.org/research**.

---

1   See the following research studies for more details:

Linda Hargrove, Donn Godin, and Barbara Dodd, *College Outcomes Comparisons by AP and Non-AP High School Experiences* (New York: The College Board, 2008).

Chrys Dougherty, Lynn Mellor, and Shuling Jian, *The Relationship Between Advanced Placement and College Graduation* (Austin, Texas: National Center for Educational Accountability, 2006).

Offering AP Courses and Enrolling Students

This *AP Course and Exam Description* details the essential information required to understand the objectives and expectations of an AP course. The AP Program unequivocally supports the principle that each school implements its own curriculum that will enable students to develop the content knowledge and skills described here.

Schools wishing to offer AP courses must participate in the AP Course Audit, a process through which AP teachers' syllabi are reviewed by college faculty. The AP Course Audit was created at the request of College Board members who sought a means for the College Board to provide teachers and administrators with clear guidelines on curricular and resource requirements for AP courses and to help colleges and universities validate courses marked "AP" on students' transcripts. This process ensures that AP teachers' syllabi meet or exceed the curricular and resource expectations that college and secondary school faculty have established for college-level courses. For more information on the AP Course Audit, visit **www.collegeboard.org/apcourseaudit**.

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 underrepresented. 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.

How AP Courses and Exams Are Developed

AP courses and exams are designed by committees of college faculty and expert AP teachers who ensure that each AP subject reflects and assesses college-level expectations. To find a list of each subject's current AP Development Committee members, please visit **apcentral.collegeboard.com/apc/public/courses/228674.html**. AP Development Committees define the scope and expectations of the course, articulating through a curriculum framework what students should know and be able to do upon completion of the AP course. Their work is informed by data collected from a range of colleges and universities to ensure that AP coursework reflects current scholarship and advances in the discipline.

The AP Development Committees are also responsible for drawing clear and well-articulated connections between the AP course and AP Exam — work that includes designing and approving exam specifications and exam questions. The AP Exam development process is a multiyear endeavor; all AP Exams undergo extensive review, revision, piloting, and analysis to ensure that questions are high quality and fair and that there is an appropriate spread of difficulty across the questions.

Throughout AP course and exam development, the College Board gathers feedback from various stakeholders in both secondary schools and higher education institutions. This feedback is carefully considered to ensure that AP courses and exams are able to provide students with a college-level learning experience and the opportunity to demonstrate their qualifications for advanced placement upon college entrance.

How AP Exams Are Scored

The exam scoring process, like the course and exam development process, relies on the expertise of both AP teachers and college faculty. While multiple-choice questions on the end-of-course exam are scored by machine, the through-course performance tasks are scored by college faculty and expert AP teachers at the annual AP Reading. AP Exam Readers are thoroughly trained, and their work is monitored throughout the Reading for fairness and consistency. In each subject, a highly respected college faculty member fills the role of Chief Reader, who, with the help of AP Readers in leadership positions, maintains the accuracy of the scoring standards. Scores on the through-course performance tasks are weighted and combined with the weighted results of the computer-scored multiple-choice questions on the end-of-course assessment and this composite score is converted into an AP Exam score of 5, 4, 3, 2, or 1.

The score-setting process is both precise and labor intensive, involving numerous psychometric analyses of the results of a specific AP Exam in a specific year and of the particular group of students who took that exam. Additionally, to ensure alignment with college-level standards, part of the score-setting process involves comparing the performance of AP students with the performance of students enrolled in comparable courses in colleges throughout the United States. In general, the AP composite score points are set so that the lowest raw score needed to earn an AP score of 5 is equivalent to the average score among college students earning grades of A in the college course. Similarly, AP Exam scores of 4 are equivalent to college grades of A−, B+, and B. AP Exam scores of 3 are equivalent to college grades of B−, C+, and C.

Using and Interpreting AP Scores

College faculty are involved in every aspect of AP, from course and exam development to scoring and standards alignment. These faculty members ensure that the courses and exams meet colleges' expectations for content taught in comparable college courses. Based on outcomes research and program evaluation, the American Council on Education (ACE) and the Advanced Placement Program recommend that colleges grant credit and/or placement to students with AP Exam scores of 3 and higher. The AP score of 3 is equivalent to grades of B−, C+, and C in the equivalent college course. However, colleges and universities set their own AP credit, advanced standing, and course placement policies based on their unique needs and objectives.

AP Score	Recommendation
5	Extremely well qualified
4	Well qualified
3	Qualified
2	Possibly qualified
1	No recommendation

Additional Resources

Visit **apcentral.collegeboard.org** for more information about the AP Program.

About the AP Computer Science Principles Course

The AP Computer Science Principles course is designed to be equivalent to a first-semester introductory college computing course. In this course, students will develop computational thinking skills vital for success across all disciplines, such as using computational tools to analyze and study data and working with large data sets to analyze, visualize, and draw conclusions from trends. The course engages students in the creative aspects of the field by allowing them to develop computational artifacts based on their interests. Students will also develop effective communication and collaboration skills by working individually and collaboratively to solve problems, and will discuss and write about the impacts these solutions could have on their community, society, and the world.

Prerequisites

It is recommended that a student in the AP Computer Science Principles course should have successfully completed a first-year high school algebra course with a strong foundation in basic linear functions and composition of functions, and problem-solving strategies that require multiple approaches and collaborative efforts. In addition, students should be able to use a Cartesian (x, y) coordinate system to represent points in a plane. It is important that students and their advisers understand that any significant computer science course builds on a foundation of mathematical and computational reasoning that will be applied throughout the study of the course.

Participating in the AP Course Audit

Schools wishing to offer AP courses must participate in the AP Course Audit. Participation in the AP Course Audit requires the online submission of two documents: the AP Course Audit form and the teacher's syllabus. The AP Course Audit form is submitted by the AP teacher and the school principal (or designated administrator) to confirm awareness and understanding of the curricular and resource requirements. The syllabus, detailing how requirements are met, is submitted by the AP teacher for review by college faculty.

Please visit **http://www.collegeboard.com/html/apcourseaudit/teacher.html** for the **Curricular and Resource Requirements** that identify the set of curricular and resource expectations that college faculty nationwide have established for a college-level course, as well as for more information and materials to support syllabus development, including:

▶ **Annotated Sample Syllabi** — Provide examples of how the curricular requirements can be demonstrated within the context of actual syllabi.

▶ **Example Textbook List** — Includes a sample of AP college-level textbooks that meet the content requirements of the AP course.

▶ **Syllabus Development Guide** — Includes the guidelines reviewers use to evaluate syllabi along with three samples of evidence for each requirement. This guide also specifies the level of detail required in the syllabus to receive course authorization.

▶ **Syllabus Development Tutorial** — Describes the resources available to support syllabus development and walks through the syllabus development guide requirement by requirement.

AP Computer Science Principles Curriculum Framework

Introduction

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 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

Based on the Understanding by Design® (Wiggins and McTighe) model, the AP Computer Science Principles curriculum framework provides a clear and detailed description of the course requirements necessary for student success. The course is designed to be equivalent to a first-semester introductory college computing course. 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: 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 (EU) is aligned with one or more **learning objectives** that provide a detailed articulation of what students are expected to be able to do by the end of the course. The learning objectives integrate a computational thinking practice or skill with specific content and provide clear information about how students will be expected to demonstrate their knowledge and abilities. They are numbered to correspond with the big ideas and enduring understandings (e.g., LO 7.2.1 is from Big Idea 7, Enduring Understanding 7.2, and it is the first learning objective in that section). **The learning objectives will be the target of assessment on the AP Computer Science Principles through-course performance tasks and AP End-Of-Course Exam**.

▸ Next to each learning objective (LO) 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, the following exclusion statement applies to Learning Objective 4.2.1:

  › **Exclusion Statement (for LO 4.2.1):** Any discussion of nondeterministic polynomial (NP) is beyond the scope of this course and the AP 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 performance tasks.

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

The two through-course performance tasks require students to explore the impacts of computing and create computational artifacts digitally and 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.

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

CURRICULUM FRAMEWORK

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 websites, and writing programs. Students in this course engage in the creative aspects of computing by designing and developing interesting computational artifacts as well as by applying computing techniques to creatively solve problems. Students are expected to:

▶ Create a computational 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 intrinsically both 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 the investigation of questions using data sets and 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.

The 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 (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
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, an audio, a 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 (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
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 1.2.1** Create a computational artifact for creative expression. [P2] (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, three-dimensional (3-D) 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 (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
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 (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
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.

Enduring Understandings (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
EU 2.1 A variety of abstractions built on binary sequences can be used to represent all digital data. (continued)	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.
		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 abstractions in developing programs help to more effectively apply available resources and tools to solve problems.

Enduring Understandings (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
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 (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
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 (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
EU 3.1 People use computer programs to process information to gain insight and knowledge.	**LO 3.1.1** 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 (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
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 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 (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
EU 3.2 Computing facilitates exploration and the discovery of connections in information. (continued)	LO 3.2.1 Extract information from data to discover and explain connections or trends. [P1] (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 Determine how large data sets impact the use of computational processes to 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.

Enduring Understandings (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
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.
		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 (Students will understand that …)	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.

Enduring Understandings (Students will understand that …)	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. (continued)	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.
		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 the number of steps the algorithm takes is less than or equal to a polynomial function (constant, linear, square, cube, etc.) of the size of the input.
		EXCLUSION STATEMENT (for EK 4.2.1B): Using nonpolynomial functions to describe relationships between the number of steps required by an algorithm and the input size is beyond the scope of this course and the AP Exam.
		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.

Enduring Understandings (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
EU 4.2 Algorithms can solve many, but not all, computational problems. (continued)	**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.
	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.

Enduring Understandings (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
EU 4.2 Algorithms can solve many, but not all, computational problems. (continued)	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 (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
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 (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
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 code segments 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 (Students will understand that …)	**Learning Objectives** (Students will be able to …)	**Essential Knowledge** (Students will know that …)
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.
		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.

Enduring Understandings (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
EU 5.3 Programming is facilitated by appropriate abstractions. (continued)	LO 5.3.1 Use abstraction to manage complexity in programs. [P3] (continued)	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 segments are harder to reason about than shorter code segments 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.
		EK 5.4.1J Justification can include a written explanation about how a program meets its specifications.

Enduring Understandings (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
EU 5.4 Programs are developed, maintained, and used by people for different purposes. (continued)	**LO 5.4.1** Evaluate the correctness of a program. [P4] (continued)	**EK 5.4.1K** Correctness of a program depends on correctness of program components, including code segments 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.
		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]	**EK 6.1.1A** The Internet connects devices and networks all over the world.
	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 6.1.1B** 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.
		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 domain 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).

Enduring Understandings (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
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.
		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.

Concept Outline: Big Idea 6

Enduring Understandings (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
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 Cyberwarfare and cybercrime 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.
		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.

CURRICULUM FRAMEWORK

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 computing innovations, which are innovations that include a computer or program code as an integral part of their function. 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.

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. (continued)	**LO 7.1.1** Explain how computing innovations affect communication, interaction, and cognition. [P4] (continued)	**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.
		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.

Enduring Understandings (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
EU 7.2 Computing enables innovation in nearly every field. (continued)	**LO 7.2.1** Explain how computing has impacted innovations in other fields. [P1] (continued)	**EK 7.2.1E** Open and curated scientific databases have benefited scientific researchers.
		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 global effects — 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 (Students will understand that …)	Learning Objectives (Students will be able to …)	Essential Knowledge (Students will know that …)
EU 7.3 Computing has global effects — both beneficial and harmful — on people and society. (continued)	LO 7.3.1 Analyze the beneficial and harmful effects of computing. [P4] (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 real-world contexts, including 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.
EU 7.5 An investigative process is aided by effective organization and selection of resources. Appropriate technologies and tools facilitate the accessing of information and enable the ability to evaluate the credibility of sources.	LO 7.5.1 Access, manage, and attribute information using effective strategies. [P1]	EK 7.5.1A Online databases and libraries catalog and house secondary and some primary sources.
		EK 7.5.1B Advance search tools, Boolean logic, and key words can refine the search focus and/or limit search results based on a variety of factors (e.g., data, peer-review status, type of publication).
		EK 7.5.1C Plagiarism is a serious offense that occurs when a person presents another's ideas or words as his or her own. Plagiarism may be avoided by accurately acknowledging sources.
	LO 7.5.2 Evaluate online and print sources for appropriateness and credibility. [P5]	EK 7.5.2A Determining the credibility of a source requires considering and evaluating the reputation and credentials of the author(s), publisher(s), site owner(s), and/or sponsor(s).
		EK 7.5.2B Information from a source is considered relevant when it supports an appropriate claim or the purpose of the investigation.

AP Computer Science Principles Instructional Approaches

The AP Computer Science Principles course provides students with the opportunity to develop computational thinking skills, an understanding of the real-world impact of computing, and programming literacy. The course exposes students to the breadth and relevance of computer science across many fields of study that incorporate computer science knowledge. A strong focus on creativity as it applies to the creation of computational artifacts allows a broader range of students to discover where computer science could fit in their lives, and it prepares more students for success in computer science and other related STEM fields.

When designing a plan to teach the course, teachers should include ample opportunities for students to collaborate and be creative as they engage in the learning of the course content. Teachers should establish protocols and use platforms that facilitate collaboration. **Cooperative-learning** strategies and **pair programming** are suggested practices for fostering collaboration and creativity.

Because the field of computer science changes rapidly, teachers should strive to keep their courses current by reserving class time to investigate emerging computing innovations that are being used for creative expression or to solve authentic problems. Whenever possible, teachers can use recent technology to help students connect the applicable nature of computing to multiple fields of study. Providing opportunities for students to explore and discuss computing innovations that impact their lives will help them develop their own ideas for creating computational artifacts.

Planning Your Course

Choosing a Programming Language

AP Computer Science Principles does not require a specific programming language. Because a goal of this course is to broaden participation, teachers can create their AP Computer Science Principles course centered around computing concepts in the curriculum framework that support the creation of exciting and relevant computational artifacts. For this reason, teachers are encouraged to select a programming language(s) that is most appropriate for their classroom and that will provide students opportunities to successfully engage with the course content. The programming language selected should contain functionality that is specified in the curriculum framework and performance tasks. Appropriate programming languages for this course are ones that allow students to create text, evaluate expressions, and use variables, conditionals, loops, lists, and procedures. Below is short list of programming languages that can be considered to use in this course. Teachers are encouraged to examine other programming languages as they plan for instruction.

Language/Product	Description
Alice	This 3-D modeling environment allows students to create and animate 3-D worlds. This environment lends itself well to creating stories and games.
App Inventor	This open-source Web application allows students to create their own applications on mobile devices.
App Lab	This is a programming environment for creating web applications with JavaScript. It allows students to develop programs and toggle back and forth between block-based and text-based programming modes.
EarSketch	This browser-based application allows students to create their own music using either JavaScript or Python.
Greenfoot	This Java IDE is designed for use in education to create two-dimensional graphic applications, such as simulations and interactive games.
Java	There are several IDEs that can be used to write in Java. The Java language allows students to create and solve problems that vary widely in difficulty.
JavaScript	This language is commonly used to create interactive effects within Web browsers.
Lego Mindstorms NXT	This product integrates programming with Lego bricks and sensors to create and program robots. The instructions are assembled by linking together function blocks.
Processing	This programming language was initially created to serve as a software sketchbook, and it can be used to teach programming using a visual context.
Python	This language has the benefit of readability that might be helpful to new programmers.
Scratch	This blocks-based programming language allows students to build scripts to run animations. This product can be downloaded and installed on a computer or run in the browser.
Snap*!*	This Scratch-style programming language is block-based and allows users to define new primitives in JavaScript. Users can read and write information from the Internet using server-defined APIs and make mobile applications.
Swift	This programming language is designed for use with iOS, OS X, tvOS and watchOS. This environment allows students to create their own Apple apps and includes interactive environments that allow students to see the effects of changes or additions to code as they type.

INSTRUCTIONAL APPROACHES

This course provides students with an AP Computer Science Principles Exam Reference Sheet, as seen in the Reproducibles for Students section, to use when taking the AP Exam. The exam reference sheet is NOT meant to be a substitute for choosing a programming language that is recognized by the field of computer science or being used in postsecondary institutions. Because there is no designated programming language for the AP Computer Science Principles course, the exam reference sheet was developed in order to be able to assess students' knowledge and skills in programming constructs as described in the curriculum framework. It is meant to establish a common way to communicate programming concepts for the purpose of the exam. Teachers are encouraged to integrate the use of the exam reference sheet throughout the school year.

Organizational Approaches

There are several different approaches for organizing an AP Computer Science Principles course. This section will explain and give examples of how to use the approaches that are represented in one or more of the course planning and pacing guides, which can be found on AP Central. As teachers are designing their course and deciding on the objectives of lessons, they may want to pose a project as a problem or as a question and allow students to create a computational artifact to solve the problem or answer the question. Providing students with opportunities to collaborate as they work through their projects while combining these approaches together creates an environment that is conducive to developing critical thinking and problem-solving skills in a creative way.

Approach	Key Characteristics	Examples in AP Computer Science Principles
Project based	▶ Students are presented with an outline for a long-term project or a problem to solve. ▶ Students are required to plan, implement, and test a solution. ▶ Students work in groups or individually to complete a project. ▶ Content can be taught as necessary or taught prior to the start of the project.	▶ Students create a story or tutorial for beginning computer science students to learn how the Internet transmits information. ▶ Students create a project to address the problem of sending a message publicly across the classroom without other students being able to discern the message.
Integrated	▶ Teachers spiral the curriculum to address multiple big ideas in the same project or unit.	Students read an article on a current technology and answer the following questions: ▶ Where are the generalizations and simplifications being made by the author? Is this abstraction? ▶ What are the potential beneficial and harmful effects of this technology? ▶ What data might be collected through the use of this technology? ▶ How would this technology use the Internet?
Inquiry based	▶ Students construct their own knowledge rather than acquiring it. ▶ New information is constructed based on personal experiences. ▶ Teachers guide students toward personal discoveries. ▶ Topics are introduced through current events and student experiences.	Teachers share an experience where they had a difficult time sending a picture through email. Students add their own knowledge and experience to the conversation and new knowledge emerges about how compression works. Some guiding questions might be: ▶ How is a picture stored on a computer? ▶ What are some ways that we can reduce the file size of a picture? ▶ What is the difference between lossy and lossless compression? ▶ Under which circumstances would you use each?

Investigations in Computer Science

The AP Computer Science Principles course and the Explore — Impact of Computing Innovations performance task require students to conduct investigations of computing innovations. A computing innovation is an innovation that includes a computer or program code as an integral part of its function. Some examples of computing innovations include:

▸ physical computing innovations such as Google glasses and self-driving cars; and

▸ non-physical computing software (such as cell phone applications) or computing concepts (such as eCommerce), which rely on physical transactions conducted on the Internet.

Information on the through-course performance tasks appears in the Assessment Overview section. Teachers should help students understand that the investigation process is not simply about collecting evidence or facts and then piecing them together. This process should include asking questions and coming up with solutions and conclusions through thoughtful reflections, and often in the case of computer science, creating a computational solution. During the investigation process, AP Computer Science Principles students seek relevant information in articles, books, and other sources as well as through examining data, and they develop an informed perspective built on the ideas in the examined materials.

Evaluation of Sources

A critical element in conducting these investigations is the evaluation of sources. Students will be expected to discern sources as credible and relevant. If available, the school media specialist can be a resource to assist students with searching school databases, determining credible sources, and appropriately citing the reliability and credibility of sources.

The following are examples of instructional strategies and learning tools that can be used to help facilitate student investigations and the evaluation of the relevance and credibility of sources:

▸ **Prompting** students to help them plan for and conduct their investigations. It is recommended to use this strategy to scaffold the performance tasks by having students regularly conduct investigations and practice using evidence from this investigation, through the use of in-text citations, when answering questions that are similar to those in the through-course performance tasks.

▸ Providing students with a **graphic organizer** that will help to discern a source's credibility based on a list of criteria about the source's reputation, expertise, and potential slant.

▸ Using **modeling** and a **think aloud** to model the evaluation of sources.

Sources should be evaluated for relevance and credibility. The following table illustrates considerations for evaluating sources.

Criteria	Considerations	AP Computer Science Principles Examples and Illustrations
Credibility	Credibility requires students to obtain information from sources with knowledge and expertise of the computing innovation. Sources can be print, online, or expert interviews. However, because many students are using the Internet or database sources, students should consider the following: ▶ The domain name extensions indicate who publishes and owns the domain. ▶ The author of the website. ▶ The sources that are cited in the materials as well as websites they link to are credible as well.	Commonly used extensions include: .edu (educational organization); .com (company); .org (any organization); .gov (government agency); .net (network). Read past the first slash / in the domain name to see if the page might be someone's personal page. A personal page might be less credible. Finding out who the author is, their credentials, and the organization they are associated with will help to determine whether they are qualified to write about the topic. An article that includes citation of resources is often more credible. Take the time to evaluate a few of the cited resources as well.
Relevance	Students should examine the content of a source (the evidence) to ensure it supports their claims and provides insight and knowledge that relates to the topic. This means that evidence is only relevant when it addresses both the topic in context and the student's argument.	An article describing the spacious seating area of the Google self-driving car may not be relevant in supporting a claim regarding the safety of the vehicle.
	Because we are investigating computing innovations that are rapidly changing, it is important to consider if the information being provided is the most recent and current. This means students should make sure they are addressing the current version of the computing innovation, as well as the most recent published information.	An article about version 4 of a computing innovation might not be relevant when the current version is version 8.

Supporting Claims With Evidence

As students investigate computing innovations, they are required to provide evidence for claims made as part of those investigations. Acknowledgments of the evidence and claims must be made through in-text citations and listing the sources of the citations as references. Whenever students are conducting their investigations or creating computational artifacts, students are required to acknowledge their sources. They are also required to acknowledge program code segments that they did not write themselves, including open-source program code segments, as well as any student conclusion that is supported by data sets, articles, books, and other sources.

When providing instruction on acknowledging sources, teachers should instruct students as follows:

▶ **Text-based sources:** Students should properly acknowledge sources that are used to provide evidence in their written responses through in-text citations, especially when quoting directly from a source. Using phrases such as "according to" and

"as seen in" or using direct quotes with parenthetical citation would be considered acceptable means of acknowledging sources as well as a way for students to build credibility. Although there is no specific citation style required in AP Computer Science Principles course, MLA, APA, or IEEE would be appropriate to use for a works-cited page.

▸ **Program code:** Students should properly acknowledge program code they did not explicitly write, such as APIs, open-source program code, or program code provided during peer-to-peer collaboration. Open source refers to access and licensing of program code that is publicly available for use or modification. One way students can acknowledge someone else's program code is by adding a comment either to the program code or to the documentation, crediting the author and listing where the program code was found. This includes program code that may have been written together as part of whole-class instruction.

▸ **Media:** Students should acknowledge any external media used in the creation of their computational artifacts. External media includes: images, music and other media that are not original student work. This media can be acknowledged on the computational artifact itself, in a credits section, or in a works-cited page.

Creative works and innovations often become intellectual property of the creators or owners and require acknowledgment. Failure to acknowledge the ideas and works of others is considered plagiarism. Teachers should inform students of the consequences of plagiarism and instruct students to ethically acknowledge, attribute, and cite the ideas and work of others. A student who fails to acknowledge the source or author of any and all information about existing program code or a computing innovation should be provided with feedback on a proper way to credit sources.

Creating Computational Artifacts

To prepare students to successfully complete the through-course performance tasks, as well as to create more authentic learning experiences, ample time should be spent in the practice of creating computational artifacts. A computational artifact is a visualization, a graphic, a program, a video, or an audio that students create using a computer. The computational artifact could solve a problem, show creative expression, or provide a viewer with new insight or knowledge.

When creating computational artifacts, teachers should encourage students to think outside of traditional means for communicating ideas (e.g., PowerPoint presentations); students should be steered toward more creative avenues.

The following are some examples of creative computational artifacts:

▸ Promotional materials, a song, or a video commercial that could be used to advertise the function or purpose of a new computing innovation

▸ An educational tutorial, an online book, or a program that teaches a younger student about something related to computer science

▸ A public service announcement or poster denouncing cyberbullying

▸ An infographic that presents and compares data related to social media

To guide students in conducting computer science investigations (e.g., investigating solutions to a problem, conducting data analysis, or analyzing computing innovations) and in creating computational artifacts, teachers can use the following strategies:

▶ **Modeling** an iterative development process for planning and creating computational artifacts.

▶ Using graphic organizers, such as **KWHL charts** and **vocabulary organizers**, when reading about current events, new technologies, or computer-based solutions to problems. A KWHL chart shows: what students **K**now; what they **W**ant to know; **H**ow they will find information; and what they have **L**earned.

▶ **Modeling** some of the available tools that can be used to create computational artifacts including programs.

▶ Scaffolding performance tasks by providing students with the opportunity to practice completing mock performance tasks that have an authentic purpose and audience. These mock performance tasks can be shorter in length, focus on different requirements than the actual performance tasks, and allow students to demonstrate a transfer of knowledge learned.

▶ Utilizing **cooperative learning strategies** and providing students with many opportunities to participate in learning activities while creating computational artifacts for a more authentic experience.

▶ Having students practice using **pair programming** when collaborating to write a program and having students reflect on this process.

Development Process

The organizational approaches in the table earlier in this section include instruction that facilitates students' use of a process to plan and create computational artifacts. Some traditional process plans that emphasize computational thinking practices include the software-development lifecycle and the engineering-design process. Planning processes are iterative and cyclical in nature and require students to reflect on what they have created. If necessary, students return to prior stages to modify their plans and change their development.
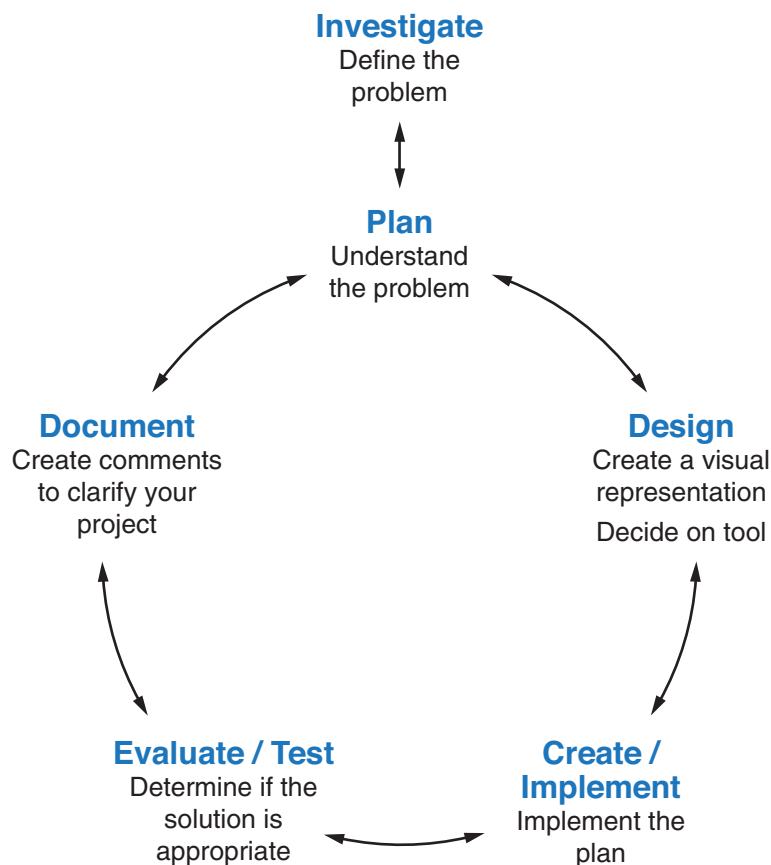
The following elements can be integrated within a development process to plan and create computational artifacts. These elements are included in both the software-development lifecycle and the engineering-design process. Throughout the use of any design process, students will be using the **computational thinking practice** of **collaborating**.

▶ **Investigate** – In this phase, students explore the **computational thinking practice** of **analyzing problems and artifacts** while they define the problem that needs to be solved.

▶ **Plan** – In this phase, students implement the **computational thinking practices** of **creating computational artifacts** and **abstraction**. Teachers can use strategies such as **create a plan**, **identify a subtask**, **look for a pattern**, and **mark the text** to help students understand the problem they need to solve.

▶ **Design** – In this phase, students create a visual representation or model of their solution and decide which tools can be used to solve the problem. They use the **computational thinking practice** of **creating computational artifacts** as they select appropriate techniques to develop their solution.

▸ **Create/Implement** – In this phase, students use the **computational thinking practice** of **collaborating** to implement their plan.

▸ **Evaluate/Test** – In this phase, students use the **computational thinking practices connecting computing** and **analyzing problems and artifacts**. Teachers can use strategies such as **predict and compare** and **error analysis** to determine whether the solution is appropriate and the potential impact of the computing innovation. This phase may require students to revisit some of the earlier phases in the development to make improvements on their product.

▸ **Document** – In this phase, students implement the **computational thinking practice** of **communicating**. When creating computational artifacts, this phase allows students to describe their process and add comments to program code. Students could keep a journal or log book to note design decisions and rationales. This allows students to come back to a computational artifact later and recall how it was constructed. This is helpful when modifications need to be made to the computational artifact.

The elements listed above need not be implemented in linear order. Students may choose to return to earlier phases as their design ideas change and develop.

The following graphic illustrates the iterative and cyclical nature of a development process to plan and create computational artifacts.

INSTRUCTIONAL APPROACHES

Investigate
Define the problem

Plan
Understand the problem

Document
Create comments to clarify your project

Design
Create a visual representation
Decide on tool

Evaluate / Test
Determine if the solution is appropriate

Create / Implement
Implement the plan

Linking Computational Thinking Practices and Learning Objectives

The following table provides examples of how to connect computational thinking practices to the learning objectives, along with suggested instructional strategies for teaching these learning objectives.

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Connecting computing [P1]	**LO 3.2.1** Extract information from data to discover and explain connections or trends.	▶ Describe the data that is being collected. ▶ What questions can be answered through analysis of this data? ▶ How is data being used by the government, businesses, and individuals?	Programming and problem-solving strategies ▶ Look for a pattern Cooperative learning ▶ Discussion group ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner Making connections ▶ Activate prior knowledge
	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. **LO 4.2.2** Explain the difference between solvable and unsolvable problems in computer science. **LO 4.2.3** Explain the existence of undecidable problems in computer science.	▶ Describe the problem in your own words. ▶ How efficient would a computing solution be for this problem? ▶ Why is the computer the correct tool to use when solving this problem? ▶ What would the estimated run time be to solve this problem?	Programming and problem-solving strategies ▶ Predict and compare ▶ Think aloud Cooperative learning ▶ Discussion group

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Connecting computing [P1] (continued)	**LO 5.5.1** Employ appropriate mathematical and logical concepts in programming.	▶ What pattern do you see in the data? ▶ How can we simplify the process through the use of an equation? ▶ What equation would be needed to find the solution?	Programming and problem-solving strategies ▶ Create a plan ▶ Identify a subtask ▶ Modeling ▶ Predict and compare ▶ Simplify the problem ▶ Think aloud Cooperative learning ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner ▶ Use manipulatives
	LO 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it.	▶ What are the risks faced when using the Internet? ▶ What privacy and security concerns should we be aware of? ▶ How can you protect yourself on the Internet? ▶ How does the Internet change the way we connect with others?	Cooperative learning ▶ Discussion group ▶ Online tools for collaboration ▶ Think-pair-share ▶ Turn to your partner Making connections ▶ Activate prior knowledge

INSTRUCTIONAL APPROACHES

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Connecting computing [P1] (continued)	**LO 7.2.1** Explain how computing has impacted innovations in other fields. **LO 7.4.1** Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts. **LO 7.5.1** Access, manage, and attribute information using effective strategies.	▶ Who does this computing innovation affect? ▶ How are they affected? ▶ Would the effect be social, economic, or cultural? Justify your response. ▶ What are the potential beneficial and harmful effects of this computing innovation? ▶ How has this computing innovation changed our lives socially, economically, and culturally? How will it? ▶ What advanced search parameters can be employed to narrow search results? ▶ What are the benefits of acknowledging sources? ▶ What is the proper way to acknowledge various sources used in written responses? In programming code? In artifacts?	Cooperative learning ▶ Discussion group ▶ Online tools for collaboration ▶ Think-pair-share ▶ Turn to your partner Making connections ▶ Activate prior knowledge ▶ KWHL chart ▶ Paraphrase

INSTRUCTIONAL APPROACHES

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Creating computational artifacts [P2]	**LO 1.1.1** Apply a creative development process when creating computational artifacts. **LO 1.2.1** Create a computational artifact for creative expression. **LO 1.2.2** Create a computational artifact using computing tools and techniques to solve a problem. **LO 1.2.3** Create a new computational artifact by combining or modifying existing artifacts. **LO 1.3.1** Use computing tools and techniques for creative expression.	▶ Why would a computational solution be appropriate to solve this problem? ▶ Which computational tools could we use? ▶ Which tool would be most appropriate and why?	Programming and problem-solving strategies ▶ Create a plan ▶ Think aloud Cooperative learning ▶ Discussion group ▶ Think pair share ▶ Turn to your partner
	LO 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts.	▶ Describe how we can break this problem into smaller, more manageable pieces.	Programming and problem-solving strategies ▶ Code tracing ▶ Create a plan ▶ Error analysis ▶ Identify a subtask ▶ Look for a pattern ▶ Marking the text ▶ Modeling ▶ Simplify the problem Cooperative learning ▶ Kinesthetic learning ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner ▶ Unplugged activities

INSTRUCTIONAL APPROACHES

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Creating computational artifacts [P2] (continued)	**LO 4.1.1** Develop an algorithm for implementation in a program.	▶ What subtasks exist for this problem? ▶ What steps would we follow to find the solution? ▶ What processes, if any, are repeated?	Programming and problem-solving strategies ▶ Create a plan ▶ Identify a subtask ▶ Look for patterns ▶ Marking the text ▶ Modeling ▶ Simplify the problem Cooperative learning ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner ▶ Unplugged activities
	LO 5.1.1 Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge. **LO 5.1.2** Develop a correct program to solve problems.	▶ What is the purpose of the program? ▶ Justify how a program would be an appropriate tool to solve the problem. ▶ What test cases can we use to determine if our program code is correct? ▶ Describe how we can break this program into more manageable pieces.	Programming and problem-solving strategies ▶ Code tracing ▶ Create a plan ▶ Identify a subtask ▶ Look for a pattern ▶ Marking the text ▶ Pair programming ▶ Simplify the problem ▶ Think aloud ▶ Work backward Cooperative learning ▶ Student response system ▶ Think-pair-share

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Abstracting [P3]	**LO 2.1.1** Describe the variety of abstractions used to represent data. **LO 2.2.2** Use multiple levels of abstraction to write programs. **LO 2.2.3** Identify multiple levels of abstractions that are used when writing programs. **LO 2.3.1** Use models and simulations to represent phenomena. **LO 2.3.2** Use models and simulations to formulate, refine, and test hypotheses.	▶ Describe how data is represented in the computer. ▶ What algorithm can we use to convert between different number systems? ▶ Describe how bits are grouped to represent numbers, characters, and colors. ▶ Describe how you can make your code more generic and reusable by including procedures, parameters, or variables. ▶ Describe how the programmer made her or his program code more reusable by including procedures, parameters, or variables. ▶ What are some models or simulations that you know? ▶ Why are models and simulations important? ▶ Describe how a model or simulation uses abstraction.	Programming and problem-solving strategies ▶ Code tracing ▶ Create a plan ▶ Identify a subtask ▶ Look for a pattern ▶ Marking the text ▶ Modeling ▶ Pair programming ▶ Simplify the problem ▶ Think aloud Cooperative learning ▶ Discussion group ▶ Kinesthetic learning ▶ Sharing and responding ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner ▶ Unplugged activities ▶ Use manipulatives Making connections ▶ Activate prior knowledge ▶ Interactive word wall ▶ Note-taking ▶ Vocabulary organizer
	LO 3.2.2 Determine how large data sets impact the use of computational processes to discover information and knowledge.	▶ What are some challenges when using large data sets? ▶ Describe how this data analysis approach will need to change as the size of the data set expands.	Programming and problem-solving strategies ▶ Create a plan ▶ Error analysis ▶ Look for pattern ▶ Modeling ▶ Predict and compare ▶ Simplify the problem Making connections ▶ Paraphrase

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Abstracting [P3] (continued)	**LO 5.2.1** Explain how programs implement algorithms. **LO 5.3.1** Use abstraction to manage complexity in programs.	▶ What subtasks, if any, can you identify in this program code? ▶ Which sections of your program code can be grouped under one name? What would you name the sections? ▶ Identify the algorithms used in this program. ▶ Identify where abstraction is being used in a program.	Programming and problem-solving strategies ▶ Code tracing ▶ Create a plan ▶ Identify a subtask ▶ Look for a pattern ▶ Marking the text ▶ Modeling ▶ Pair programming ▶ Simplify the problem ▶ Think aloud ▶ Work backward Cooperative learning ▶ Discussion group ▶ Kinesthetic learning ▶ Sharing and responding ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner ▶ Unplugged activities ▶ Use manipulatives
	LO 6.1.1 Explain the abstractions in the Internet and how the Internet functions.	▶ Describe how the Internet deals with the large diversity of devices connected to it. ▶ What system of protocols exists so that devices can communicate via the Internet? ▶ Describe how new devices are connected to the Internet.	Cooperative learning ▶ Discussion group ▶ Kinesthetic learning ▶ Sharing and responding ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner ▶ Unplugged activities ▶ Use manipulatives Making connections ▶ Interactive word wall ▶ KWHL chart ▶ Paraphrase ▶ Note-taking ▶ Vocabulary organizer

INSTRUCTIONAL APPROACHES

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Analyzing problems and artifacts [P4]	**LO 1.2.5** Analyze the correctness, usability, functionality, and suitability of computational artifacts.	▶ What are the similarities and differences in the proposed solutions? ▶ Describe how one solution would be more efficient than another. ▶ When and why is efficiency important?	Programming and problem-solving strategies ▶ Code tracing ▶ Error analysis ▶ Modeling ▶ Predict and compare ▶ Work backward Cooperative learning ▶ Discussion group ▶ Sharing and responding ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner ▶ Unplugged activities
	LO 3.1.1 Find patterns and test hypotheses about digitally processed information to gain insight and knowledge. **LO 3.3.1** Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information.	▶ Describe how computers are used in data analysis. ▶ What are the different ways we can represent data? ▶ Compare different representations of the same set of data. What patterns emerge? ▶ Determine whether the different representations are accurate. ▶ Compare the conclusions made by different representations.	Programming and problem-solving strategies ▶ Look for a pattern ▶ Modeling ▶ Pair programming ▶ Predict and compare Cooperative learning ▶ Discussion group ▶ Sharing and responding ▶ Think-pair-share ▶ Turn to your partner Making connections ▶ Activate prior knowledge ▶ Online tools for collaboration ▶ Paraphrase

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Analyzing problems and artifacts [P4] (continued)	**LO 4.2.4** Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity.	▶ What input values will cause this algorithm to produce inaccurate results? ▶ What is another way to solve this problem? Compare the efficiency of the two solutions. ▶ Compare multiple algorithms that produce the same result. How would the run times of each algorithm vary? ▶ When does efficiency become important? ▶ What is the difference between a linear search and a binary search in terms of process and efficiency?	Programming and problem-solving strategies ▶ Error analysis ▶ Look for a pattern ▶ Modeling ▶ Predict and compare ▶ Simplify the problem ▶ Work backward Cooperative learning ▶ Discussion group ▶ Kinesthetic learning ▶ Sharing and responding ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner ▶ Unplugged activities ▶ Use manipulatives
	LO 5.4.1 Evaluate the correctness of a program.	▶ What test cases can be used to evaluate this program?	Programming and problem-solving strategies ▶ Code tracing ▶ Error analysis ▶ Look for a pattern ▶ Modeling ▶ Predict and compare ▶ Simplify the problem ▶ Work backward Cooperative learning ▶ Discussion group ▶ Sharing and responding ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner ▶ Unplugged activities ▶ Use manipulatives

INSTRUCTIONAL APPROACHES

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Analyzing problems and artifacts [P4] (continued)	**LO 6.2.2** Explain how the characteristics of the Internet influence the systems built on it.	▶ What design decisions were made when building the Internet? ▶ What is the role of redundancy on the Internet? ▶ Describe how information is sent via the Internet. ▶ Describe how size and speed of systems affect their use.	Cooperative learning ▶ Discussion group ▶ Kinesthetic learning ▶ Online tools for collaboration ▶ Sharing and responding ▶ Think-pair-share ▶ Turn to your partner ▶ Unplugged activities ▶ Use manipulatives Making connections ▶ Activate prior knowledge ▶ Interactive word wall ▶ KWHL chart ▶ Note-taking ▶ Paraphrase ▶ Vocabulary organizer
	LO 7.1.1 Explain how computing innovations affect communication, interaction, and cognition. **LO 7.1.2** Explain how people participate in a problem-solving process that scales. **LO 7.3.1** Analyze the beneficial and harmful effects of computing.	▶ Describe how current forms of communication have changed, both positively and negatively, the way we interact with one another. ▶ Describe how data is being used. How is it being used positively? Negatively? ▶ Describe how we are using technology to foster collaboration. Give an example. ▶ What is the legal way to leverage open-source material?	Cooperative learning ▶ Discussion group ▶ Online tools for collaboration ▶ Sharing and responding ▶ Think-pair-share ▶ Turn to your partner ▶ Use manipulatives Making connections ▶ Activate prior knowledge ▶ KWHL chart ▶ Paraphrase

INSTRUCTIONAL APPROACHES

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Communicating [P5]	**LO 2.1.2** Explain how binary sequences are used to represent digital data.	▸ Describe how the use of data is limited. ▸ What can we use bits to represent?	Cooperative learning ▸ Discussion group ▸ Online tools for collaboration ▸ Sharing and responding ▸ Think-pair-share ▸ Turn to your partner Making connections ▸ Activate prior knowledge ▸ Interactive word wall ▸ KWHL chart ▸ Note-taking ▸ Paraphrase ▸ Vocabulary organizer
	LO 3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notations, and precise language.	▸ What data can be used to support your claims? ▸ What are the different visual ways we can represent this data? ▸ Which representation would be most effective in communicating your findings and conclusions?	Programming and problem-solving strategies ▸ Modeling Cooperative learning ▸ Discussion group ▸ Kinesthetic learning ▸ Online tools for collaboration ▸ Sharing and responding ▸ Think-pair-share ▸ Turn to your partner ▸ Unplugged activities ▸ Use of manipulatives Making connections ▸ Activate prior knowledge ▸ Paraphrase

INSTRUCTIONAL APPROACHES

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Communicating [P5] (continued)	**LO 4.1.2** Express an algorithm in a language.	▶ What are some of the ways that we can express an algorithm? ▶ Which algorithm language should we use to express an algorithm? Justify your answer.	Programming and problem-solving strategies ▶ Create a plan ▶ Marking the text ▶ Modeling ▶ Think aloud Cooperative learning ▶ Discussion group ▶ Kinesthetic learning ▶ Sharing and responding ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner ▶ Unplugged activities ▶ Use manipulatives Making connections ▶ Activate prior knowledge
	LO 6.2.1 Explain characteristics of the Internet and the systems built on it.	▶ What are the different parts of a domain name? ▶ What are the different parts of an IP address?	Cooperative learning ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner ▶ Use manipulatives Making connections ▶ Activate prior knowledge ▶ Interactive word wall ▶ KWHL chart ▶ Note-taking ▶ Vocabulary organizer

INSTRUCTIONAL APPROACHES

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Communicating [P5] (continued)	LO 7.5.2 Evaluate online and print sources for appropriateness and credibility.	▶ How is the information in the resource relevant to your investigation? ▶ Justify whether or not the content is accurate. ▶ What is the perspective of the author? ▶ What bias did the author have when writing this source? ▶ What makes this source credible? ▶ To what extent was the author able to observe or have access to credible evidence? ▶ What is the purpose of the website? ▶ What authorship clues does the URL provide (e.g., .com, .edu, .gov)? ▶ What are the qualifications of the author or group that created the website? ▶ When was the website last revised, modified, or updated? ▶ Is the website well maintained? ▶ What is your opinion of the appearance of the website?	Investigation ▶ Discussion group ▶ Graphic organizer ▶ Modeling ▶ Paraphrase ▶ Prompting ▶ Think aloud ▶ Think-pair-share ▶ Turn to your partner

INSTRUCTIONAL APPROACHES

Computational Thinking Practices	Where the Computational Thinking Practice Is Addressed in the Curriculum Framework	Questioning and Instructional Cues	Instructional Strategies (Descriptions of these strategies can be found on pages 61–64.)
Collaborating [P6]	**LO 1.2.4** Collaborate in the creation of computational artifacts.	▶ Describe how you plan to divide your work. ▶ Who will be responsible for completing which pieces? ▶ Describe how you will share your work during the collaborative process. ▶ Describe how you will balance the use of resources.	Programming and problem-solving strategies ▶ Create a plan ▶ Identify a subtask Cooperative learning ▶ Discussion group ▶ Online tools for collaboration ▶ Sharing and responding ▶ Think-pair-share ▶ Turn to your partner Making connections ▶ Paraphrase
	LO 3.1.2 Collaborate when processing information to gain insight and knowledge.	▶ Describe how you plan to divide your work. ▶ Who will be responsible for completing which pieces? ▶ Describe how you will share your work during the collaborative process. ▶ Describe how you will balance the use of resources.	Cooperative learning ▶ Discussion group ▶ Online tools for collaboration ▶ Sharing and responding ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner Making connections ▶ Paraphrase
	LO 5.1.3 Collaborate to develop a program.	▶ Describe how you plan to divide your work. ▶ Who will be responsible for completing which pieces? ▶ Describe how you will share your work during the collaborative process. ▶ Describe how you will balance the use of resources.	Programming and problem-solving strategies ▶ Create a plan ▶ Error Analysis ▶ Identify a subtask ▶ Modeling ▶ Pair programming Cooperative learning ▶ Discussion group ▶ Online tools for collaboration ▶ Sharing and responding ▶ Student response system ▶ Think-pair-share ▶ Turn to your partner Making connections ▶ Paraphrase

INSTRUCTIONAL APPROACHES

Representative Instructional Strategies

The table below contains suggested instructional strategies that can be used when teaching computer science. Instructional strategies have been categorized based on whether they are used to teach programming or problem solving, are suitable for cooperative learning, are helpful for students to make connections between material being presented and prior knowledge of topics covered, or provide guidance for teaching students how to conduct investigations.

Category	Instructional Strategy	Definition	Application to Computer Science Principles
Programming and problem-solving strategies	Code tracing	Students step through program code by hand to determine how a piece of program code operates.	These strategies are intended to help students manage tasks, encourage students to persevere in the development of a program, and build students' confidence as the types of problems they are asked to solve become more complex.
	Create a plan	Students analyze the tasks in a problem and create a process for completing the tasks by finding the information needed, interpreting data, choosing how to solve a problem, communicating results, and verifying accuracy.	▶ Students can do **pair programming** as a means for collaboration. One way to conduct pair programming is to have students switch roles every 20 minutes.
	Error analysis	Students analyze an existing solution to determine whether (or where) errors have occurred.	▶ Teachers can use **identify a subtask** to help students break programs into procedures. This is an ideal situation to teach the incorporation of abstraction into programming.
	Identify a subtask	Students break a problem into smaller pieces whose outcomes lead to a solution.	
	Look for a pattern	Students observe trends by organizing data or creating representations.	▶ Having students **look for a pattern**, **work backward**, **simplify the problem**, **create a plan**, **mark the text**, and **think aloud** are all useful for students who are having difficulty approaching a problem.
	Marking the text	Students highlight, underline, and/or annotate text to focus on key information to help understand the text or solve the problem.	
	Modeling	A teacher demonstrates a new concept or approach to learning during which students learn by observing.	▶ A critical part of learning to successfully create programs is for students to learn how to use **error analysis** and **code tracing** to determine if there are any issues with their programs. Students can practice by collaborating with a peer and examining the program code of a peer.
	Pair programming	Two programmers work together as a pair. One (the driver) writes program code, while the other (the observer, pointer, or navigator) reviews each line of program code as it is typed in.	
	Predict and compare	Students make conjectures about what results will develop in an activity, confirming or modifying the conjectures based on outcomes.	
	Simplify the problem	Students use friendlier numbers to solve a problem.	▶ Students can also check their program code by using **predict and compare** to see if their program code works as anticipated.
	Think aloud	Students talk through a difficult problem by describing what the text means.	

Category	Instructional Strategy	Definition	Application to Computer Science Principles
Programming and problem-solving strategies (continued)	Work backward	Students trace a possible answer back through the solution process to the starting point.	▸ Teachers can use **modeling** as a way of demonstrating how to construct algorithms and solve problems. They can also model a **think aloud** and vocalize their thought process for students.
Cooperative learning	Discussion group	Students engage in an interactive, small-group discussion.	Cooperative learning is an educational approach that aims to organize classroom activities into academic and social learning experiences. Many of these strategies can be used together to create a cooperative learning environment for increased student learning and engagement.
	Kinesthetic learning	The learner's body movements are used to create knowledge or understanding of a new concept. A kinesthetic–tactile learning style requires students to manipulate or touch materials to learn.	▸ **Kinesthetic learning** and **unplugged activities** can be effective ways to help students understand what they cannot see happening inside a computer. Student learning and engagement can be increased through acting out the way a computer operates.
	Sharing and responding	Students communicate with another person or a small group of peers who respond to a proposed problem or solution.	
	Student response system	Students use a classroom response system or other means of electronic or nonelectronic communication to send answers or information in response to a teacher's question or request.	
	Think-pair-share	Students think through a problem alone, pair with a partner to share ideas, and then share results with the class.	▸ Using **think-pair-share** and **turn to your partner** are helpful strategies when reviewing information from a prior day's lesson as well as when connecting new material to existing knowledge.
	Turn to your partner	Students share a formulated individual response with a partner.	
	Unplugged activities	Students use engaging games and puzzles that use manipulatives and kinesthetic-learning activities.	▸ Teachers can use a **student response system** to conduct formative assessments and receive feedback from students.
	Use manipulatives	Students use objects to examine relationships between the information given.	

INSTRUCTIONAL APPROACHES

Category	Instructional Strategy	Definition	Application to Computer Science Principles
Making connections	Activate prior knowledge	The teacher provides students an opportunity to recall what they already know about a concept and make connections to current studies.	Providing students an opportunity to make sense of computer science terms is essential, especially if the terms are new to students.
	Interactive word wall	Students use an interactive visual display of vocabulary words as a constant reminder of words and groups of words as they are introduced, used, and mastered over the course of a year.	▶ Teachers can use scaffolding and **activate prior knowledge** to provide students with a base on which to build their knowledge. Teachers can scaffold learning to reinforce topics and deepen knowledge.
	KWHL chart	Students use a graphic organizer that allows them to activate prior knowledge by identifying what they know, identifying what they want to know, identifying how they will acquire this knowledge and reflecting on what they have learned.	▶ The use of graphic organizers such as **KWHL charts** and **vocabulary organizers** as well as **note-taking** and **interactive word walls** can be very beneficial strategies in building students' vocabulary.
	Note-taking	Students create a record of information while listening to a speaker.	▶ When reading current event articles, teachers can use a **KWHL chart** to help students connect what they are reading to what they already know about a technology.
	Paraphrase	Students restate in their own words essential information expressed in text.	▶ Teachers can asks students to **paraphrase** information they have read about a computing innovation.
	Vocabulary organizer	Students use a graphic organizer with a designated format to maintain an ongoing record of vocabulary words with definitions, pictures, notation, and connections.	
Investigation	Discussion group	Students engage in an interactive, small-group discussion.	Students will be required to conduct computer science investigations on solutions to problems, data analysis, or analyzing computing innovations and need to develop the skills required to evaluate sources as credible and relevant. These strategies can assist students in determining the credibility of sources.
	Online tools for collaboration	Students use online resources that allow them to participate in discussions and collaborate on a common project. This is helpful in facilitating the division of assigned work among partners or groups of students.	▶ Teachers can use **online tools for collaboration** and **sharing and responding** to help facilitate **discussion groups** while researching solutions to problems and analyzing data and information.
	Graphic organizer	Students use a visual representation for the organization of information.	▶ Teachers can use **modeling** and a **think aloud** to demonstrate for students how to evaluate the reliability and credibility of a source.
	Modeling	A teacher demonstrates a new concept or approach to learning during which students learn by observing.	
	Paraphrase	Students restate in their own words essential information expressed in text.	

Category	Instructional Strategy	Definition	Application to Computer Science Principles
Investigation (continued)	Prompting	Students use written prompts as a guide to systematically look for evidence that supports or refutes key claims.	▶ Teachers can provide a **graphic organizer** to help students discern a source's credibility based on a list of criteria about the source.
	Think aloud	Students talk through a difficult problem by describing what the text means.	▶ Teachers can use cooperative learning techniques such as **discussion group**, **think-pair-share**, and **turn to your partner** to build student understanding when validating sources.
	Think-pair-share	Students think through a problem alone, pair with a partner to share ideas, and then share results with the class.	
	Turn to your partner	Students share a formulated individual response with a partner.	▶ Teachers can use **prompting** to provide students with a problem or purpose for their investigations and scaffold performance tasks.
			▶ Teachers can ask students to **paraphrase** to ensure they understand the problem or desired need and are not distorting information based on their own biases.

Linking Course Concepts and Strategies

This section highlights some of the computational thinking practices and big ideas that are fundamental to the study of computer science by providing strategies that will help engage and motivate students.

Using Strategies for Collaboration

Collaboration is clearly an important part of this course — it is, after all, one of the computational thinking practices. During the development of computing innovations, collaboration allows computer scientists to improve their products. Collaboration takes place in a variety of ways:

▶ brainstorming ideas and solutions in a team environment;

▶ working together to design subtasks of a larger project, developing these subtasks, and then integrating them in the completion of the project;

▶ providing feedback on a computational artifact, including a program, to improve the overall quality; and

▶ providing technical support when problems arise that an individual is struggling to solve on their own.

Students will need many opportunities to practice collaboration throughout the course, especially as they consider different collaborative methods that will help them complete the Create — Applications from Ideas performance task.

*Some effective ways to incorporate collaboration in your classroom include:*

▶ The cooperative learning strategies that are outlined in the instructional strategies table above can be used to help foster collaborative relationships in the classroom. Having an established protocol for students to have equal participation and share their ideas, such as the **think-pair-share** strategy, builds students' confidence and can create a successful collaborative learning community within a classroom.

▶ Teachers should provide students with opportunities to work together to solve problems. River-crossing problems and critical thinking problem-solving questions can be solved collaboratively. It is useful to form student groups of two to three, assigning groups in various ways and mixing groups often. A strong emphasis should be placed on valuing and discussing contributions of ideas from all group members and obtaining consensus on the approach(es) to solving a problem.

*The following are examples of different forms of collaborating while programming:*

▶ Collaboration may involve soliciting feedback from a collaborative partner at various points in their program development and providing feedback to a collaborative partner(s) at various points in their program development.

▶ Collaboration may involve each collaborative partner developing pieces of the program, combining those pieces, and providing feedback during the development process.

▶ Collaboration can take the form of brainstorming and sharing ideas before the process of writing program code begins. Collaborative partners can then choose to work together or independently at selected times during the programming process.

▶ Collaboration can take the form of working together to develop an idea, beginning the programming process together, and then working independently to add different features to the collaboratively developed portion of the program.

▶ Collaboration can employ *pair programming*, in which one collaborative partner "drives" (enters code) while the other "navigates" (recommends and reviews program code entered by driver), with the collaborative partners changing roles after designated time intervals. For multiday projects, teachers can have students rotate roles every 20–30 minutes. Smaller programming projects can be used as warm-up problems where students rotate roles with each warm-up.

▶ Collaboration can blend any or all of the above techniques and may include an iterative process in which one or more of these techniques, or other collaboration techniques, are employed several times in the program design, implementation, and testing phases.

The illustrations below demonstrate steps in the software development life cycle and show two of many possible approaches:

1. Students may collaboratively work on a single program based on a single idea, iterating through the usual stages of the software development life cycle, and then separate to add their individual part to it.
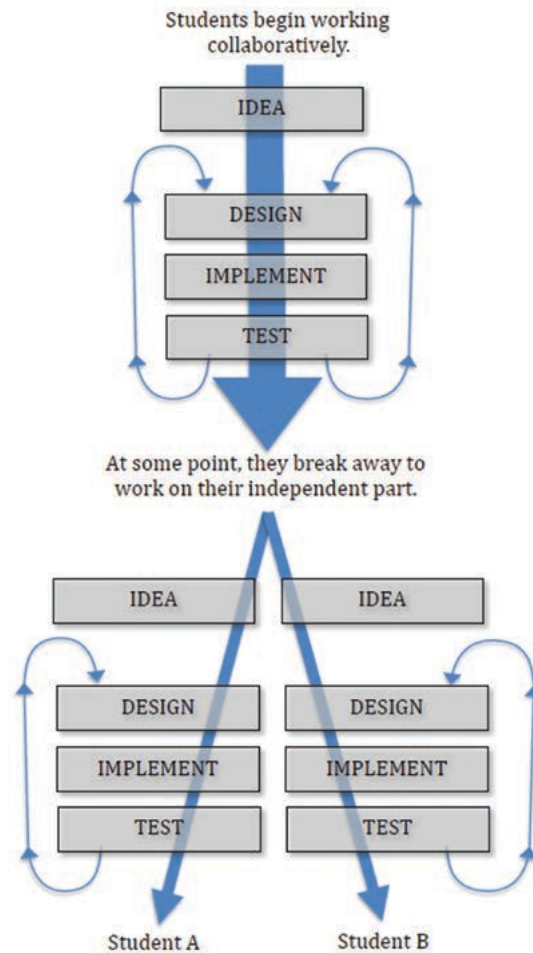
2. Students may start out on their own, each generating independent ideas. Then, they can collaborate at different points during the development process.
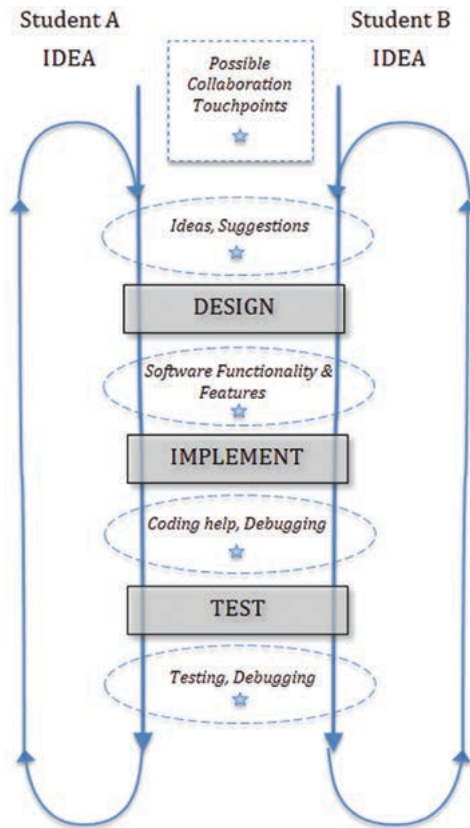
The course emphasizes collaboration as a way for students to work with a partner to brainstorm solutions, successfully accomplish a task, and improve upon the quality of work. Learning to collaborate includes understanding different perspectives and approaches to solving problems. One way to assess how well students collaborate is by having them write reflections or participate in discussions about how collaborating with a partner helped accomplish their task or improve the quality of their computational artifact. Students should be asked to reflect not only on what went well in the collaboration process but also where they could improve it. Some partnerships require more intervention from the teacher than others. When a partnership is not working well, mediation and reflection can be helpful in getting the partnership back on track. The reflection should focus on a student's own actions and how he or she works with others to solve problems. Teachers should guide students to share their reflections and consider different ways their work can be improved as they continue to collaborate.

Introducing Abstraction

The topic of abstraction takes on many forms for computer science. Abstraction occurs when we simplify a process by creating a generalization. It is seen in the way computers store data, with the lowest level being bits; seen in programs through the use of variables, procedures, and parameters; and used in simulations or models to isolate the features that need to be looked at more closely.

Abstraction can be categorized as either data abstraction or procedural abstraction.

▶ **Data Abstraction:** When students are introduced to the concept of data abstraction, it is hard for them to comprehend that everything stored on a computer is just bits. Their pictures, music, emails, and book reports are stored on a computer as bits. Teachers are encouraged to look up online resources that can be employed to help students understand how a computer works and stores data. To introduce students to the data abstraction, teachers can:

› have students read an article on how computers store data and use a making-connections strategy, such as **activate prior knowledge**, **KWHL chart**, or **note-taking**; and

› show students the ASCII chart that is used to map the characters on the keyboard to numbers and **model** how to encode messages using the decimal or octal equivalents for the letters.

Now that students have had an introduction to how data is stored, it is time they understand how the binary number system works. To accomplish this, teachers can:

› use the **activate prior knowledge** strategy to build on students' existing knowledge of how the decimal number system works; and

› apply this knowledge of place value to building the place values for the binary, octal, and hexadecimal number systems.

▶ **Procedural Abstraction:** In terms of programming, an outcome of the use of procedural abstraction is program code that is more readable and reusable. Through the use of variables, algorithms, and procedures, program code is broken into subtasks that are more easily managed. To demonstrate this for students, teachers can:

› use programming and problem-solving strategies such as **simplify the problem**, **identify a subtask**, or **create a plan** to break programs down into individual, reusable tasks;

› have students practice the software design cycle to plan and develop a solution that includes abstraction;

› have students **think aloud**, **mark the text**, and **look for patterns** (patterns often indicate the use of procedural abstraction);

› use **kinesthetic learning** and **unplugged activities**, having students form groups to play a board game and identify the different abstractions in the forms of procedures and variables;

› use **modeling** to demonstrate how to write procedures, continuing to model the various levels of abstraction by adding parameters and variables;

› provide students with example pieces of program code and have them use **error analysis** to identify the abstractions in the program code and then use the **turn to your partner** strategy;

› use **code tracing** or **work backward** to understand how the overall program interacts with its abstract components; and

› have students collaborate by use **pair programming** to improve provided program code by adding abstraction.

Investigating Data and Information

Data and information are very important in the field of computer science. Many of our computer systems are heavily reliant on the collection and use of data to determine important information or new knowledge. It is what makes simulations more robust; for example, it is what ensures that you get the right coupons when checking out at the grocery store, and it is what helps you navigate through many of the apps on today's technological devices. Much of this data is used in ways that mean it is transformed into information that can impact our everyday actions and transactions. Data is also often stored for potential future use. As students learn about the existence of data and how it can be used and transformed into information, their understanding can help them see the connection between computer science and other fields of study and careers. To help students understand this about data and information, teachers can:

▶ have students use **KWHL charts** to investigate the ways in which they generate data and make predictions as to how this data could be utilized;

▶ in **discussion groups** or with **online tools for collaboration**, have students read about and examine the possible uses of data in various fields, such as medicine, business, criminal justice, marketing, civil engineering, and municipal planning;

▶ have students generate and pose questions about a set of data, and then use **sharing and responding** to refine questions;

▶ have students use **predict and compare** and **error analysis** while analyzing data — this will help them identify anomalies within a data set; and

▶ use **student response systems** for students to generate their own data for analysis.

An important aspect of this course is the flexibility to use a variety of computing tools to teach many of the concepts in the curriculum framework. To provide students an opportunity to discover information and knowledge while working with data, teachers can use spreadsheets and other databases to analyze data. Teachers can also have students develop a program to process information from the data they have explored. The data and/or information can be used to create multiple representations, such as graphs, tables, diagrams, and even pictures and sounds. The different ways that the data and information can be represented can help to provide new insight or solve problems.

Teaching the Internet

The Internet is a global communication network that society has become very dependent on, but students are not necessarily aware of how it works. Characteristics of the Internet and cybersecurity issues are important aspects computer scientists consider when solving problem and creating software or new computing innovations. Teachers can consider using online resources to help students understand Internet protocol and allow them to explore the history and interworking of the Internet. To begin teaching the Internet, teachers can:

▶ use a making-connections strategy such as **paraphrase** or a **vocabulary organizer** to help students synthesize what they are reading in a technical text;

▶ use an **interactive word wall** to facilitate the learning of new Internet vocabulary;

▶ follow up any reading with **discussion group** or **note-taking** to ensure students of different reading abilities all have the correct information; and

▶ **use manipulatives** and **kinesthetic-learning** activities to illustrate how information is sent via packets through the Internet.

Additional Resources

The resources listed below provide additional information on some instructional strategies included in this course and exam description. The list also includes resources for the design and development of projects and programming code. Teachers can use and integrate information from resources when planning for instruction and are encouraged to investigate other appropriate resources as well.

Classroom Instruction that Works

Robert J. Marzano, Jana S. Marzano, and Debra J. Pickering. *Classroom Instruction that Works: Research-based Strategies for Increasing Student Achievement*. 2001. Association for Supervision and Curriculum Development.

Cer B. Dean, Elizabeth R. Hubbell, Howard Pitler, and Bj Stone. *Classroom Instruction that Works: Research-based Strategies for Increasing Student Achievement*. 2012. Association for Supervision and Curriculum Development.

Collaborative Development

"Collaborative Learning: Group Work." *Cornell University Center for Teaching Excellence*. **http://www.cte.cornell.edu/teaching-ideas/engaging-students/collaborative-learning.html**

"4 Methods to Enhance Student Collaboration in the Classroom." *Concordia Portland Online*. **http://education.cu-portland.edu/blog/reference-material/4-methods-to-enhance-student-collaboration-in-the-classroom/**

Computer Science Education

Computer Science Teachers Association Resources: **http://www.csteachers.org/?page=Resources**

Jane Margolis, Rachel Estrella, Joanna Goode, Jennifer Jellison Holme and Kim Nao. *Stuck in the Shallow End: Education, Race, and Computing*. 2010. The MIT Press.

Cooperative Learning

David W. Johnson, Roger T. Johnson, and Edythe Johnson Holubec. "The New Circles of Learning: Cooperation in the Classroom and School." 1994. Association for Supervision and Curriculum Development. **http://www.ascd.org/publications/books/194034/chapters/What-Is-Cooperative-Learning%C2%A2.aspx**

Murphy, Laurie, Kenneth Blaha, Tammy VanDeGrift, Seven Wolfman, and Carol Zander. "Active and Cooperative Learning Techniques for The Computer Science Classroom." Dec. 2002. Web. Dec. 2015. **http://faculty.up.edu/vandegri/tenure/papers/ccscnw02/p92-murphy.pdf**

Design and Development Process

"What Is the Software Development Life Cycle?" *Official Blog Airbrake Bug Tracker*. **https://airbrake.io/blog/insight/what-is-the-software-development-life-cycle**

"Engineering Design Process." **https://www.teachengineering.org/engrdesignprocess.php**

"The Engineering Design Process." **http://www.eie.org/overview/engineering-design-process**

Mohammed, Nabil, Ali Munassar, and A. Govardhan. "A Comparison Between Five Models of Software Engineering." *IJCSI International Journal of Computer Science* 7.5 (2010): 94-101. **http://www.ijcsi.org/papers/7-5-94-101.pdf**

KWHL Chart

"KWHL Chart." *KWHL Chart*. **http://www.graphic.org/kwhl.html**

Pair Programming

Williams, Laurie, Robert R. Kessler, Ward Cunningham, and Ron Jeffries. "Strengthening the Case for Pair Programming." *IEEE Software* July/August (2000): 19-25. *IEEE Software*. IEEE, July 2000. Web. Dec. 2015. **http://collaboration.csc.ncsu.edu/laurie/Papers/ieeeSoftware.PDF**

Project Based Learning

"What Is Project Based Learning (PBL)?" *What Is PBL?* **http://bie.org/about/what_pbl**

Prompting

"Strategy of the Week: Writing Prompts." *Education World* **http://www.educationworld.com/a_curr/strategy/writing_prompts.shtml**

Understanding by Design

Grant P. Wiggins, Jay McTighe. *Understanding by Design*. 2005. Association for Supervision and Curriculum Development.

Open Source

"What Is Open Source?" *Opensource.com*. **https://opensource.com/resources/what-open-source**

Open Source Initiative. **http://opensource.org/**

AP Computer Science Principles Assessment Overview

The AP Computer Science Principles course has three assessments consisting of two performance tasks and an end-of-course AP Exam. All of these assessments are summative and the scoring results from each will be used to calculate a final AP score using the 1–5 scale. Each assessment will count for a certain percentage of the total AP score, as shown in the table below.

Assessment	Timing	Percentage of Total AP Score
Explore Performance Task	8 hours	16%
Create Performance Task	12 hours	24%
End-of-Course Exam	2 hours	60%

The AP Computer Science Principles through-course performance tasks allow students to demonstrate proficiency in course content and skills that cannot be assessed on the end-of-course AP Exam, such as creating a computational artifact. One of the tasks requires students to develop a program and the other to investigate a computing innovation and create a computational artifact that represents the computing innovation's intended purpose, function, or its effect. A computing innovation is an innovation that includes a computer or program code as an integral part of its functionality. Student work will be submitted online to the College Board through the AP Digital Portfolio. Instructions for online submissions are available on the AP Computer Science Principles Course Home Page.

The AP Computer Science Principles End-of-Course Exam is 2 hours long. It is a paper and pencil exam and includes 74 multiple-choice questions. There are two types of multiple-choice questions:

▶ **Single-select multiple-choice questions:** Students select one answer from among four options.

▶ **Multiple-select multiple-choice questions:** Students select two answers from among four options.

The following overview provides general guidelines for the through-course performance tasks as well as the role of the teacher administering the tasks. A student version of the performance tasks appears in the Reproducibles for Students section. The student version includes guidelines and instructions for completing the performance tasks and should be distributed to students. The performance tasks can be completed in any order.

Preparing for the Through-Course Performance Tasks

General Guidelines

Prior to beginning a performance task:

▸ Provide instruction and practice, along with feedback, related to content and skills that will help students succeed on the performance tasks (e.g., abstraction and algorithm development, various types of computing innovations, unfamiliar terms).

▸ Provide multiple opportunities to practice and discuss either the entire performance tasks or individual prompts of each of the tasks. Teachers should explain the role (described in the Overview of Performance Task sections for each performance task) that they can play in providing assistance during the actual performance task; they should encourage students to take advantage of the opportunity to get assistance and feedback during practice.

▸ Review the scoring guidelines with students to help them understand how their work will be assessed. Teachers should remind students that the scoring guidelines are closely related with the prompts in the performance tasks, so they must respond to all of the prompts in their attempt to obtain the highest score possible.

▸ Provide examples of performance task submissions at high, medium, and low levels according to the scoring guidelines to understand the performance expectations. Examples of responses to each performance task can be found on AP Central. If students select a computing innovation or a program that has been used as an example or was discussed in class, students must find new sources and submit original responses to avoid plagiarism. Students cannot submit any work from AP Central samples or practice performance tasks for their final submission.

▸ Provide explicit instructions about the file submission requirements to the AP Digital Portfolio.

▸ Provide information regarding the evaluation and acknowledgment of a source including program code and media.

▸ Suggest a timeline and schedule for students to complete the performance tasks.

▸ Clarify the requirements and prompts in each performance task when students do not understand the directions.

AP Computer Science Principles Policy on Plagiarism

A student who fails to acknowledge (i.e., through citation, through attribution, by reference, and/or through acknowledgment in a bibliographic entry) the source or author of any and all information or evidence taken from the work of someone else will receive a score of 0 on that performance task.

To the best of their ability, teachers will ensure that students understand ethical use and acknowledgment of the ideas and work of others, as well as the consequences of plagiarism. The student's individual voice should be clearly evident, and the ideas of others must be acknowledged, attributed, and/or cited.

Performance Task: Explore – Impact of Computing Innovations

Weight: 16% of the AP Computer Science Principles final score

Hours: a minimum of 8 class hours

Recommended Completion Date: April 15

Submission Deadline: April 30

Note: Teachers must carefully plan a calendar that provides time for all the tasks to be completed and uploaded by April 30. Student handouts of this performance task are available in the Reproducibles for Students section.

Task Overview

Computing innovations impact our lives in ways that require considerable study and reflection for us to fully understand them. In this performance task, students will explore a computing innovation of their choice. The close examination of a computing innovation will deepen the students' understanding of computer science principles.

Components

The following components are formally assessed and must be submitted for the Explore performance task by April 30:

▶ Computational Artifact (CA)

▶ Written Responses (WR)

Learning Objectives Assessed in Explore Performance Task

	Learning Objectives for Explore Performance Task	**CA**	**WR**
1.2.1*	Create a computational artifact for creative expression. [P2]	X	
1.2.2**	Create a computational artifact using computing tools and techniques to solve a problem. [P2]	X	X
3.3.1	Analyze how data representations, storage, security, and transmission of data involve computational manipulation of information. [P4]		X
7.1.1	Explain how computing innovations affect communication, interaction, and cognition. [P4]		X
7.3.1	Analyze the beneficial and harmful effects of computing. [P4]		X
7.4.1	Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts. [P1]		X
7.5.2	Evaluate online and print sources for appropriateness and credibility. [P5]		X

  * Learning Objective 1.2.1 is assessed if students decide to create an artifact for the purpose of self-expression.

** Learning Objective 1.2.2 is assessed if students decide to create an artifact to solve a problem.

Task Guidelines

In this task, students select and investigate a computing innovation that has had or has impacts on society, economy, or culture, and consumes, produces, and/or transforms data. A computing innovation is an innovation that includes a computer or program code as an integral part of its function. Some examples of computing innovations include:

▸ physical computing innovations such as Google glasses and self-driving cars; and

▸ non-physical computing software (such as cell phone applications) or computing concepts (such as eCommerce), which rely on physical transactions conducted on the Internet.

As students analyze a computing innovation, they will explain its intended purpose or function, describe harmful and beneficial effects, describe data storage, data privacy, and data security concerns. Students will also produce a computational artifact that illustrates, represents, or explains the computing innovation's intended purpose, its function, or its effect, and provide written responses to each of the given prompts.

There are a number of widely available computational tools students can use to create computational artifacts for this task. A computational artifact is a visualization, a graphic, a video, a program, or an audio recording that students create using a computer. The students' creations could solve a problem, show creative expression, or provide the viewer with new insight or knowledge. Students must be able to attest that their computational artifact and written responses they are submitting are their own. Students can use work that is not originally created by them (including but not limited to images, video, or program code segments) as long as they provide appropriate acknowledgments.

Effective artifacts include:

▸ visual, graphical, and/or audio content to help a reader understand the purpose, function, or effect of a computing innovation; and

▸ the use of communications media, such as animations, comic strips, infographics, and/or public service announcements, to illustrate the purpose, function, or effect of a computing innovation.

Ineffective artifacts include:

▸ artifacts that repeat information supplied in the written responses;

▸ multislide presentations with paragraphs of text or bullets; and

▸ artifacts that have not been created by the student.

Students will demonstrate their understanding of computer science through the investigation and analysis of their chosen computing innovation (an innovation that has a computer or program code as an integral part of its function) and its impacts on the economy, society, and culture. Their investigations will be a thorough analysis of the function of the computing innovation based on its intended use. It is important for students to be able to explain the beneficial and harmful effects of the intended use of a computing innovation, as well as the characteristics of data input, data output, and data transformation that may occur with the computing innovation. In this way, students will make connections between important

computer science concepts and the impacts of the selected computing innovation. As students complete the Explore performance task, they must provide evidence of the extensive knowledge of computer science they have obtained throughout the course and through their investigations, and their responses should be understandable to someone who is unfamiliar with the computing innovation.

Written responses must be based on relevant, credible, and easily accessible sources. Students are required to provide in-text citations for at least three sources that helped them create their computational artifact and/or formulate their written responses. At least two of the sources must be available online or in print; the third source may be either online, in print, or a personal interview with an expert on the computing innovation. At least two of the sources must have been created after the end of the previous academic year. Students must avoid plagiarism by acknowledging, attributing, and/or citing sources throughout their responses and including a bibliography. Sources that should be acknowledged include text, images, video, music, graphs, and program code that are used in the creation of their computational artifacts.

Preparing for the Task

*Teachers **should**:*

▶ Discuss the criteria for a well-chosen computing innovation. A computing innovation is an innovation that includes a computer or program code as an integral part of its functionality. Inform students that a computing innovation that has a meaningful personal or community emphasis is an appropriate choice, as long as it fulfills all the requirements of the performance task.

▶ Provide instruction and multiple opportunities for students to practice searching and evaluating sources relevant to computing innovations. Students can search for print or nonprint sources as part of their investigation.

▶ Provide examples that appropriately illustrate or represents the purpose or function of the computing innovation.

▶ Guide students in clearly explaining the impact of a computing innovation on society, economy, and culture, clearly justifying both beneficial and harmful effects.

▶ Provide students with the meaning and purpose of creating a computational artifact. A computational artifact is a visualization, a graphic, a video, a program, or an audio recording that students create using a computer. The creation of an artifact could solve a problem, show creative expression, or provide the viewer with new insight or knowledge.

▶ Discuss the computational tools that can be used to create effective computational artifacts.

▶ Discuss computational artifacts that are effective and ineffective.

▶ Instruct students to complete an analysis of their computing innovation, including the impact the intended use of the computing innovation has on society, economy, or culture.

▶ Instruct students to demonstrate their knowledge of computer science and understanding of how data is input, output, and transformed in their analysis of the data used by the computing innovation.

▸ Instruct students to make connections between the data used by a computing innovation and a security, privacy, or storage concern.

▸ Instruct students to ensure their written responses are based on relevant and credible sources. In addition, students should ensure appropriate in-text citation of sources being quoted in a written response. Students can acknowledge information they investigate from a journal, Web page, or expert that is being quoted as part of a written response.

▸ Instruct students to ensure appropriate acknowledgment of sources used in the creation of their computational artifact. Sources that should be cited include images, video, music, written works, graphs, and program code that are used in the creation of their computational artifact.

Administering the Task: Role of the Teacher

*Teachers **must***:

▸ provide a minimum of 8 classroom hours to complete this performance task; and

▸ ensure students are aware of the performance task directions (found in the Reproducibles for Students section), timeline, and scoring criteria.

▸ allow students' interests to drive their choice of computing innovation;

▸ assist in resolving technical problems that impede work, such as a failing workstation or difficulty with access to networks, or to help with saving files;

▸ wait until after students' performance tasks have been completed and submitted as final to the AP Digital Portfolio before providing feedback on those tasks if they are being considered as part of the class grade;

▸ advise students that they may not revise their work once they have completed and submitted their work as final to the AP Digital Portfolio; and

▸ inform students they should be applying the computer science knowledge they've obtained throughout the course and in completing the performance tasks to their responses to all the prompts in the performance tasks.

*Teachers **may not***:

▸ allow students to collaborate on the Explore performance task;

▸ assign, provide, or distribute specific topics to students;

▸ write, revise, amend, or correct student work;

▸ allow students to submit computational artifacts from practice performance tasks as a submission for AP scoring;

▸ conduct or provide research, articles, and/or evidence for students; or

▸ suggest answers or provide feedback on answers to prompts.

*Teachers **may***:

▸ suggest a timeline and schedule for students for completing the performance task and monitor students' progress;

▸ clarify the requirements and prompts in each performance task when students do not understand the directions;

▸ designate consecutive or non-consecutive class hours to complete the performance task;

▸ continue whole class teaching of course content and skills during non designated time to complete the performance task;

▸ assist students in defining their focus and choice of topics prior to them beginning their investigation without making selections for them (e.g., by asking questions); students who select a program or computing innovation that was discussed in class must find new sources and submit original responses to avoid plagiarism. Students cannot submit any work from AP Central or from practice performance tasks.

▸ inform students that the scoring process that occurs in the AP Reading is different from the one that may be used in a classroom setting; the AP score that students receive may be different than their classroom grade.

▸ review the files submitted to ensure the files are correct and not corrupted for each performance task (i.e., check that the students uploaded the correct files for the Explore performance task and the correct files for the Create performance task).

Performance Task: Create – Applications From Ideas

Weight: 24% of the AP Computer Science Principles final score

Hours: a minimum of 12 class hours

Recommended Completion Date: April 15

Submission Deadline: April 30

Note: Teachers must carefully plan a calendar that provides time for all the tasks to be completed and uploaded by April 30. Student handouts of this performance task are available in the Reproducibles for Students section.

Task Overview

Programming is a collaborative and creative process that brings ideas to life through the development of software. Programs can help solve problems, enable innovations, or express personal interests. In this performance task, students will be developing a program of their choice. The students' development process should include iteratively designing, implementing, and testing their program. Students are strongly encouraged to work with another student in their class.

Students completing the AP Computer Science Principles course in a nontraditional classroom situation (e.g., online, homeschool, independent study) are permitted to collaborate with another secondary level student peer. However, a significant portion of the computer program must be developed independently. Students must provide program code segments that they developed independently as part of their written response. Students must be able to attest to the originality of the program code and the written response they are submitting. Students can use program code segments that are not originally developed by them provided they have included appropriate acknowledgment for these code segments.

There is no designated programming language for AP Computer Science Principles. Students may choose a programming language learned while taking this course to complete the task, or they may select a different programming language — one they are familiar with from outside of class. When selecting a programming language and their program focus, students should ensure that their program will be sophisticated enough to integrate mathematical and logical concepts, develop abstractions, and implement algorithms.

Components

The following components are formally assessed and must be submitted for the Create performance task by April 30:

▸ A video of your program running (V)

▸ Individual written responses about your program and development process (IWR)

▸ Program code (PC)

Learning Objectives Assessed in Create Performance Task

	Learning Objectives for Create Performance Task	V	IWR	PC
2.2.1	Develop an abstraction when writing a program or creating other computational artifacts. [P2]		X	X
4.1.1	Develop an algorithm for implementation in a program. [P2]		X	X
4.1.2	Express an algorithm in a language. [P5]		X	X
5.1.1*	Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge. [P2]	X	X	X
5.1.2**	Develop a correct program to solve problems. [P2]	X	X	X
5.2.1	Explain how programs implement algorithms. [P3]		X	
5.3.1	Use abstraction to manage complexity in programs. [P3]		X	X
5.4.1	Evaluate the correctness of a program. [P4]	X	X	X
5.5.1	Employ appropriate mathematical and logical concepts in programming. [P1]			X

  * Learning Objective 5.1.1 is assessed if students decide to create a program for the purpose of self-expression.

** Learning Objective 5.1.2 is assessed if students decide to create a program to solve a problem.

Task Guidelines

This performance task requires students to develop a program on a topic that interests them or solves a problem. It is strongly recommended that a portion of the program involve some form of collaboration with another student in their class in the planning, designing or testing (debugging) parts of the process. The program development must also involve a significant amount of independent work writing program code.

Students are required to:

▶ iteratively design, implement, and test their program;

▶ independently create at least one algorithm and one abstraction that are central to the purpose of the program and that can be used to meet the requirements for the written response;

▶ create a video that displays the running of their program and demonstrates its functionality;

▶ write responses to questions about their program to demonstrate their understanding of programming concepts; and

▸ submit their entire program code to the AP Digital Portfolio.

▸ submit their program code as a PDF file. If the program code is text based, students can use the print command to save their program code as a PDF file, or they can copy and paste the program code into a text document that can then be converted or saved as a PDF file. Students using a block-based programming language may need to use screen capture software to capture their code and paste it into a text document to then be converted into a PDF file. Students should capture clear images of only the program code.

▸ cite any program code they did not author. Program code can be cited by adding comments to the code. If the programming language does not allow for comments to be added directly to the program code, students should add them into the PDF file that contains their program code.

In this task, students develop a program that demonstrates a variety of capabilities and implements several different language features that, when combined, produce a result that cannot easily be accomplished without computing tools and techniques. The program should draw upon mathematical and logical concepts, such as use of numbers, variables, mathematical expressions with arithmetic operators, logical and Boolean operators and expressions, decision statements, iteration, and/or collections.

The program must demonstrate:

▸ use of several effectively integrated mathematical and logical concepts, from the language being used;

▸ implementation of algorithms that integrate other algorithms and mathematical and/or logical concepts; and

▸ development and use of abstractions to manage the complexity of their program (e.g., procedures, abstractions provided by the programming language, APIs).

Preparing for the Task

*Teachers **should**:*

▸ Brainstorm problems that programming can address, or brainstorm special interests that programming can help develop;

▸ Instruct students on an iterative development process. An example of the development process can be found in the Instructional Approaches section;

▸ Instruct students on the value of collaboration and different ways to collaborate with their peers;

▸ Work with students to analyze program code and code segments and explain the function as it relates to the overall program;

▸ Encourage students to keep a programming journal of the design choices that were made during the development of the program code or code segment and the effect of these decisions on the programs function. Students can use this journal as a point of reference when they are demonstrating their understanding of how:

  › an algorithm was built as part of the integration of two or more algorithms;

  › a program functions differently with the inclusion of algorithms and abstractions;

  › the inclusion of an abstraction has made their program code more compact, readable or reusable and how the program would operate differently without the inclusion of the abstraction;

▸ Instruct students to ensure appropriate acknowledgment of program code used in the creation of their computer program that is not their own. Any program code which has not been written by the student including the use of APIs, open-source code, as well as code provided during peer-to-peer collaboration, should be acknowledged and credit should be given to the author. If the programming environment allows students to include comments, this is the preferred way to acknowledge and give credit to another author. In the case of programming environments which do not have this type of functionality, students can include comments when they capture their program code for submission. Provide guidance to student as to how they can use existing program code by integrating it into their own program code or extending it in some new way.

Administering the Task: Role of the Teacher

*Teachers **must***:

▸ provide a minimum of 12 classroom hours to complete this task; and

▸ ensure students are aware of the performance task directions (found in the Reproducibles for Students section), timeline, and scoring criteria;

▸ allow students to collaborate during the development of the program code, if the student chooses to do so. During the development of the program, each student must develop their own abstraction(s) and algorithm(s) to be provided as part of their response;

▸ allow students' interests to drive their choice of projects and programming language;

▸ assist in resolving technical problems that impede work, such as a failing workstation or difficulty with access to networks, or to help with saving files;

▸ wait until after students' performance tasks have been completed and submitted as final to the AP Digital Portfolio before providing feedback on those tasks if they are being considered as part of the class grade;

▸ advise students that they may not revise their work once they have completed and submitted their work as final to the AP Digital Portfolio;

▸ inform students they should be applying the computer science knowledge they've obtained throughout the course and in completing the performance tasks to their responses to all the prompts in the performance tasks; and

- instruct students how to capture their program code to submit for the performance tasks.
  - › With text-based program code, students can use the print command to save their program code as a PDF file, or copy and paste their code to a text document and then convert it into a PDF file.
  - › With block-based program code, students can create clear screen captures that include only their program code, paste these images into a document, and then convert this document to a PDF file.

*Teachers **may not**:*

- assign, provide, or distribute to students specific topics or a program students are to develop;
- write, revise, amend, or correct student work, including debugging the program, writing or designing functionality in the program, testing the program, or making revisions to the program;
- allow students to submit computational artifacts from practice performance task as a submission for AP assessment scoring;
- suggest answers or provide feedback on answers to prompts; or
- allow students to collaborate during the creation of their video or completing written responses.

*Teachers **may**:*

- oversee the formation of groups;
- suggest a timeline and schedule for students for completing the performance task and monitor students' progress;
- clarify the requirements and prompts in each performance task when it is clear students do not understand the directions;
- designate consecutive or non-consecutive class hours to complete the performance task;
- continue whole class teaching of course content and skills during non-designated time to complete the performance task;
- assist students in defining their focus and choice of topics prior to beginning their work without making selections for them (e.g., by asking questions);
- resolve collaboration issues where one collaborative partner is clearly and directly impeding the completion of the performance tasks;
- inform students that the scoring process that occurs in the AP Reading is different from the one that may be used in a classroom setting; the AP score that students receive may be different than their classroom grade.
- review the files submitted to ensure the files are correct and not corrupted for each performance task (i.e., check that students uploaded the correct files for the Explore performance task and the correct files for the Create performance task).

AP Computer Science Principles End-of-Course Exam

Weight: 60% of the AP Computer Science Principles final score

Hours: 2 hours

Date: May (in the AP Exam administration window)

Note: The end-of-course exam will be administered using the same procedures and guidelines as all other AP Exams.

Overview

The AP Computer Science Principles End-of-Course Exam is composed of two types of multiple-choice questions:

- **Single-select multiple-choice questions:** Students select one answer from among four options.
- **Multiple-select multiple-choice questions:** Students select two answers from among four options.

Multiple-choice questions on the exam are classified according to learning objectives within each big idea in the AP Computer Science Principles curriculum framework. Some exam questions may be aligned to more than one learning objective. For example, a question on programming might implement an algorithm and contain abstractions. In any case, a primary learning objective is identified and is used to ensure the appropriate distribution of test questions in the AP Exam. The table below intends to show the approximate percentages of test question per big idea. Teachers should examine this table as one of many other important features of the course to plan their instruction.

Big Ideas	Approximate Percentage of Multiple-Choice Questions
Big Idea 1: Creativity	---
Big Idea 2: Abstraction	19%
Big Idea 3: Data and Information	18%
Big Idea 4: Algorithms	20%
Big Idea 5: Programming	20%
Big Idea 6: The Internet	13%
Big Idea 7: Global Impact	10%

The AP Computer Science Principles Exam Reference Sheet is found in the Reproducibles for Students section and provides programming instructions and explanations to help students understand questions they will see on the AP Exam.

ASSESSMENT OVERVIEW

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 knowledge of the big ideas and enduring understandings. They may address content from more than one essential knowledge statement. Exam questions may be accompanied by nontextual 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 End-Of-Course Exam and serve as examples of the types of questions that will appear on the assessment. Each question is accompanied by a table containing the enduring understandings, learning objective, computational thinking practices, and essential knowledge statements that the question addresses. The answers can be found in a table after the sample exam questions.

1. A video-streaming Web site uses 32-bit integers to count the number of times each video has been played. In anticipation of some videos being played more times than can be represented with 32 bits, the Web site is planning to change to 64-bit integers for the counter. Which of the following best describes the result of using 64-bit integers instead of 32-bit integers?

   (A) 2 times as many values can be represented.

   (B) 32 times as many values can be represented.

   (C) 2^{32} times as many values can be represented.

   (D) 32^2 times as many values can be represented.

Enduring Understandings	Learning Objectives	Computational Thinking Practices	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]	**P3** Abstracting	**2.1.1A** **2.1.1B** **2.1.1E**

2. A programmer completes the user manual for a video game she has developed and realizes she has reversed the roles of goats and sheep throughout the text. Consider the programmer's goal of changing all occurrences of "goats" to "sheep" and all occurrences of "sheep" to "goats." The programmer will use the fact that the word "foxes" does not appear anywhere in the original text.

   Which of the following algorithms can be used to accomplish the programmer's goal?

   (A) First, change all occurrences of "goats" to "sheep."
       Then, change all occurrences of "sheep" to "goats."

   (B) First, change all occurrences of "goats" to "sheep."
       Then, change all occurrences of "sheep" to "goats."
       Last, change all occurrences of "foxes" to "sheep."

   (C) First, change all occurrences of "goats" to "foxes."
       Then, change all occurrences of "sheep" to "goats."
       Last, change all occurrences of "foxes" to "sheep."

   (D) First, change all occurrences of "goats" to "foxes."
       Then, change all occurrences of "foxes" to "sheep."
       Last, change all occurrences of "sheep" to "goats."

Enduring Understandings	Learning Objectives	Computational Thinking Practices	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.	**4.1.1** Develop an algorithm for implementation in a program. [P2]	**P2** Creating computational artifacts	**4.1.1A** **4.1.1B**

3. ASCII is a character-encoding scheme that uses a numeric value to represent each character. For example, the uppercase letter "G" is represented by the decimal (base 10) value 71. A partial list of characters and their corresponding ASCII values are shown in the table below.

Decimal	ASCII Character		Decimal	ASCII Character
65	A		78	N
66	B		79	O
67	C		80	P
68	D		81	Q
69	E		82	R
70	F		83	S
71	G		84	T
72	H		85	U
73	I		86	V
74	J		87	W
75	K		88	X
76	L		89	Y
77	M		90	Z

ASCII characters can also be represented by hexadecimal numbers. According to ASCII character encoding, which of the following letters is represented by the hexadecimal (base 16) number 56?

(A) A

(B) L

(C) V

(D) Y

Enduring Understandings	Learning Objectives	Computational Thinking Practices	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]	**P3** Abstracting	**2.1.1A**
			2.1.1C
			2.1.1D
			2.1.1E
			2.1.1G

4. The figure below shows a circuit composed of two logic gates. The output of the circuit is `true`.
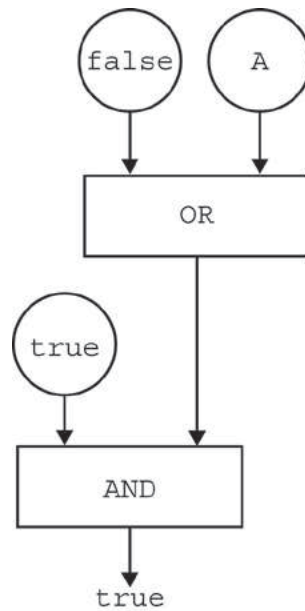
Which of the following is a true statement about input `A`?

(A) Input `A` must be `true`.

(B) Input `A` must be `false`.

(C) Input `A` can be either `true` or `false`.

(D) There is no possible value of input `A` that will cause the circuit to have the output `true`.

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
2.2 Multiple levels of abstraction are used to write programs or to create other computational artifacts.	**2.2.3** Identify multiple levels of abstractions being used when writing programs. [P3]	**P3** Abstracting	**2.2.3E** **2.2.3F**

5. The following question uses a robot in a grid of squares. The robot is represented as a triangle, which is initially in the bottom left square of the grid and facing right.
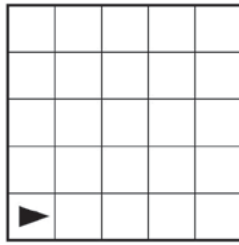
Consider the following code segment, which moves the robot in the grid.

```
n ← 3

REPEAT 3 TIMES

    REPEAT n TIMES

        MOVE_FORWARD

    ROTATE_LEFT
    n ← n - 1
```
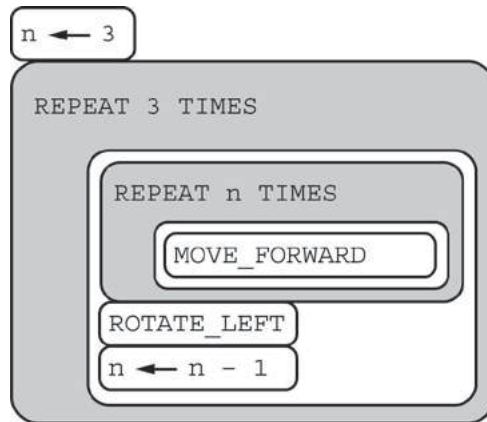
Which of the following shows the location of the robot after running the code segment?
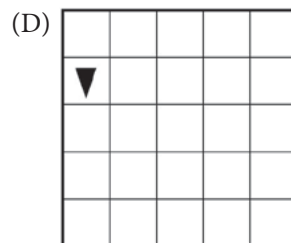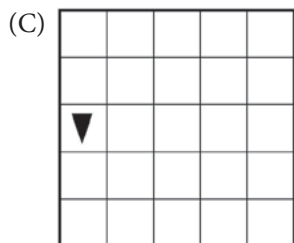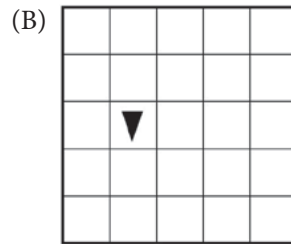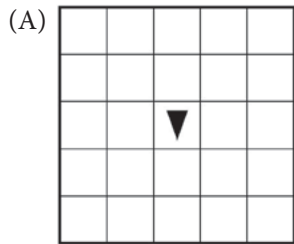
(A)

(B)

(C)

(D)

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
5.2 People write programs to execute algorithms.	**5.2.1** Explain how programs implement algorithms. [P3]	**P3** Abstracting	**5.2.1A** **5.2.1B** **5.2.1C**

6. Which of the following statements describes a limitation of using a computer simulation to model a real-world object or system?

    (A) Computer simulations can only be built after the real-world object or system has been created.

    (B) Computer simulations only run on very powerful computers that are not available to the general public.

    (C) Computer simulations usually make some simplifying assumptions about the real-world object or system being modeled.

    (D) It is difficult to change input parameters or conditions when using computer simulations.

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
2.3 Models and simulations use abstraction to generate new understanding and knowledge.	**2.3.1** Use models and simulations to represent phenomena. [P3]	**P3** Abstracting	**2.3.1A** **2.3.1C** **2.3.1D**

7. A certain social media Web site allows users to post messages and to comment on other messages that have been posted. When a user posts a message, the message itself is considered data. In addition to the data, the site stores the following metadata.

    ▪ The time the message was posted

    ▪ The name of the user who posted the message

    ▪ The names of any users who comment on the message and the times the comments were made

For which of the following goals would it be more useful to analyze the data instead of the metadata?

(A) To determine the users who post messages most frequently

(B) To determine the time of day that the site is most active

(C) To determine the topics that many users are posting about

(D) To determine which posts from a particular user have received the greatest number of comments

Enduring Understandings	Learning Objectives	Computational Thinking Practices	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 or trends. [P1]	**P1** Connecting computing	**3.2.1B** **3.2.1G** **3.2.1H** **3.2.1I**

8. The code segment below is intended to move a robot in a grid to a gray square. The program segment uses the procedure GoalReached, which evaluates to true if the robot is in the gray square and evaluates to false otherwise. The robot in each grid is represented as a triangle and is initially facing left. The robot can move into a white or gray square but cannot move into a black region.

```
REPEAT UNTIL (GoalReached ())
{
   IF (CAN_MOVE (forward))
   {
      MOVE_FORWARD ()
   }
   IF (CAN_MOVE (right))
   {
      ROTATE_RIGHT ()
   }
   IF (CAN_MOVE (left))
   {
      ROTATE_LEFT ()
   }
}
```

For which of the following grids does the code segment NOT correctly move the robot to the gray square?

(A)

(B)

(C)

(D)

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
4.2 Algorithms can solve many, but not all, computational problems.	**4.2.4** Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity. [P4]	**P4** Analyzing problems and artifacts	**4.2.4B**

9. The table below shows the time a computer system takes to complete a specified task on the customer data of different-sized companies.

Task	Small Company (approximately 100 customers)	Medium Company (approximately 1,000 customers)	Large Company (approximately 10,000 customers)
Backing up data	2 hours	20 hours	200 hours
Deleting entries from data	100 hours	200 hours	300 hours
Searching through data	250 hours	300 hours	350 hours
Sorting data	0.01 hour	1 hour	100 hours

Based on the information in the table, which of the following tasks is likely to take the longest amount of time when scaled up for a very large company of approximately 100,000 customers?

(A) Backing up data

(B) Deleting entries from data

(C) Searching through data

(D) Sorting data

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
3.2 Computing facilitates exploration and the discovery of connections in information.	**3.2.2** Determine how large data sets impact the use of computational processes to discover information and knowledge. [P3]	**P3** Abstracting	**3.2.2E** **3.2.2F** **3.2.2H**

10. Consider the code segment below.

```
IF (onTime)
    DISPLAY "Hello."
ELSE
    IF (absent)
        DISPLAY "Is anyone there?"
    ELSE
        DISPLAY "Better late than never."
```
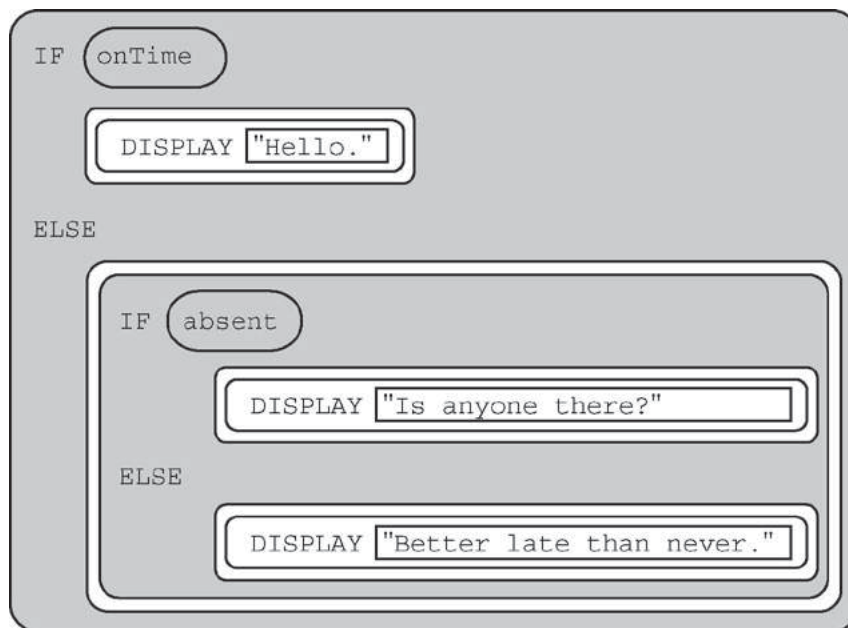
If the variables `onTime` and `absent` both have the value `false`, what is displayed as a result of running the code segment?

(A) `Is anyone there?`

(B) `Better late than never.`

(C) `Hello. Is anyone there?`

(D) `Hello. Better late than never.`

Enduring Understandings	Learning Objectives	Computational Thinking Practices	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.	**4.1.1** Develop an algorithm for implementation in a program. [P2]	**P2** Creating computational artifacts	**4.1.1A** **4.1.1C**

11. Under which of the following conditions is it most beneficial to use a heuristic approach to solve a problem?

(A) When the problem can be solved in a reasonable time and an approximate solution is acceptable

(B) When the problem can be solved in a reasonable time and an exact solution is needed

(C) When the problem cannot be solved in a reasonable time and an approximate solution is acceptable

(D) When the problem cannot be solved in a reasonable time and an exact solution is needed

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
4.2 Algorithms can solve many, but not all, computational problems.	**4.2.2** Explain the difference between solvable and unsolvable problems in computer science. [P1]	**P1** Connecting computing	**4.2.2A** **4.2.2B** **4.2.2C**

12. Which of the following are true statements about digital certificates in Web browsers?

   I. Digital certificates are used to verify the ownership of encrypted keys used in secured communication.

   II. Digital certificates are used to verify that the connection to a Web site is fault tolerant.

(A) I only

(B) II only

(C) I and II

(D) Neither I nor II

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
6.3 Cybersecurity is an important concern for the Internet and the systems built on it.	**6.3.1** Identify existing cybersecurity concerns and potential options that address these issues with the Internet and the systems built on it. [P1]	**P1** Connecting computing	**6.3.1H** **6.3.1L** **6.3.1M**

13. There are 32 students standing in a classroom. Two different algorithms are given for finding the average height of the students.

Algorithm A

Step 1: All students stand.

Step 2: A randomly selected student writes his or her height on a card and is seated.

Step 3: A randomly selected standing student adds his or her height to the value on the card, records the new value on the card, and is seated. The previous value on the card is erased.

Step 4: Repeat step 3 until no students remain standing.

Step 5: The sum on the card is divided by 32. The result is given to the teacher.

Algorithm B

Step 1: All students stand.

Step 2: Each student is given a card. Each student writes his or her height on the card.

Step 3: Standing students form random pairs at the same time. Each pair adds the numbers written on their cards and writes the result on one student's card; the other student is seated. The previous value on the card is erased.

Step 4: Repeat step 3 until one student remains standing.

Step 5: The sum on the last student's card is divided by 32. The result is given to the teacher.
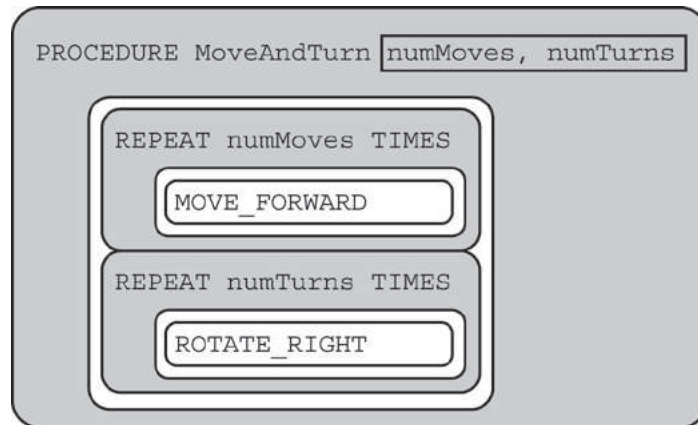
Which of the following statements is true?

(A) Algorithm A always calculates the correct average, but Algorithm B does not.

(B) Algorithm B always calculates the correct average, but Algorithm A does not.

(C) Both Algorithm A and Algorithm B always calculate the correct average.

(D) Neither Algorithm A nor Algorithm B calculates the correct average.

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
4.2 Algorithms can solve many, but not all, computational problems.	**4.2.4** Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity. [P4]	**P4** Analyzing problems and artifacts	**4.2.4C**

14. The figure below shows a robot in a grid of squares. The robot is represented as a triangle, which is initially facing upward. The robot can move into a white or gray square but cannot move into a black region.
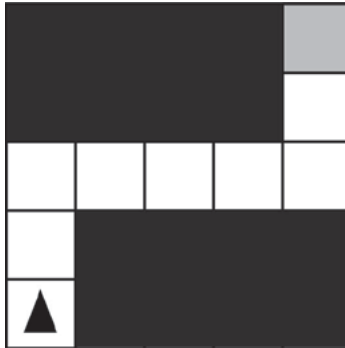
Consider the procedure `MoveAndTurn` below.

```
PROCEDURE MoveAndTurn  numMoves, numTurns

    REPEAT numMoves TIMES

        MOVE_FORWARD

    REPEAT numTurns TIMES

        ROTATE_RIGHT
```

Which of the following code segments will move the robot to the gray square?

(A)
```
MoveAndTurn 1, 2
MoveAndTurn 3, 4
MoveAndTurn 0, 2
```

(B)
```
MoveAndTurn 2, 1
MoveAndTurn 4, 1
MoveAndTurn 2, 0
```

(C)
```
MoveAndTurn 2, 1
MoveAndTurn 4, 3
MoveAndTurn 2, 0
```

(D)
```
MoveAndTurn 3, 1
MoveAndTurn 5, 3
MoveAndTurn 3, 0
```

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
5.3 Programming is facilitated by appropriate abstractions.	**5.3.1** Use abstraction to manage complexity in programs. [P3]	**P3** Abstracting	**5.3.1A** **5.3.1B** **5.3.1D** **5.3.1E** **5.3.1F** **5.3.1G**

15. Biologists often attach tracking collars to wild animals. For each animal, the following geolocation data is collected at frequent intervals.

   ▪ The time

   ▪ The date

   ▪ The location of the animal

   Which of the following questions about a particular animal could NOT be answered using <u>only</u> the data collected from the tracking collars?

   (A) Approximately how many miles did the animal travel in one week?

   (B) Does the animal travel in groups with other tracked animals?

   (C) Do the movement patterns of the animal vary according to the weather?

   (D) In what geographic locations does the animal typically travel?

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
3.1 People use computer programs to process information to gain insight and knowledge.	**3.1.1** Find patterns and test hypotheses about digitally processed information to gain insight and knowledge. [P4]	**P4** Analyzing problems and artifacts	**3.1.1E**

16. A summer camp offers a morning session and an afternoon session. The list `morningList` contains the names of all children attending the morning session, and the list `afternoonList` contains the names of all children attending the afternoon session.

   Only children who attend both sessions eat lunch at the camp. The camp director wants to create `lunchList`, which will contain the names of children attending both sessions.

The following code segment is intended to create `lunchList`, which is initially empty. It uses the procedure `IsFound (list, name)`, which returns `true` if `name` is found in `list` and returns `false` otherwise.

```
FOR EACH child IN morningList
{
   <MISSING CODE>
}
```

Which of the following could replace `<MISSING CODE>` so that the code segment works as intended?

(A)
```
IF (IsFound (afternoonList, child))
{
   APPEND (lunchList, child)
}
```

(B)
```
IF (IsFound (lunchList, child))
{
   APPEND (afternoonList, child)
}
```

(C)
```
IF (IsFound (morningList, child))
{
   APPEND (lunchList, child)
}
```

(D)
```
IF ((IsFound (morningList, child)) OR
    (IsFound (afternoonList, child)))
{
   APPEND (lunchList, child)
}
```

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
5.3 Programming is facilitated by appropriate abstractions.	**5.3.1** Use abstraction to manage complexity in programs. [P3]	**P3** Abstracting	**5.3.1G** **5.3.1K** **5.3.1L**

17. Consider the following program code.

```
i ← 0
sum ← 0

REPEAT UNTIL ( i = 4 )

        i ← 1
        sum ← sum + i
        i ← i + 1

DISPLAY sum
```
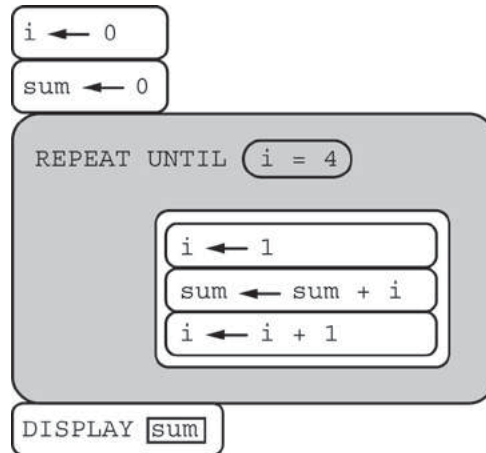
Which of the following best describes the result of running the program code?

(A) The number 0 is displayed.

(B) The number 6 is displayed.

(C) The number 10 is displayed.

(D) Nothing is displayed; the program results in an infinite loop.

Enduring Understandings	Learning Objectives	Computational Thinking Practices	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]	**P4** Analyzing problems and artifacts	5.4.1E 5.4.1K 5.4.1N

18. Which of the following is a true statement about data compression?

(A) Data compression is only useful for files being transmitted over the Internet.

(B) Regardless of the compression technique used, once a data file is compressed, it cannot be restored to its original state.

(C) Sending a compressed version of a file ensures that the contents of the file cannot be intercepted by an unauthorized user.

(D) There are trade-offs involved in choosing a compression technique for storing and transmitting data.

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
3.3 There are trade-offs when representing information as digital data.	**3.3.1** Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information. [P4]	**P4** Analyzing problems and artifacts	**3.3.1C** **3.3.1D** **3.3.1E**

19. An office building has two floors. A computer program is used to control an elevator that travels between the two floors. Physical sensors are used to set the following Boolean variables.

Variable	Description
onFloor1	set to `true` if the elevator is stopped on floor 1; otherwise set to `false`
onFloor2	set to `true` if the elevator is stopped on floor 2; otherwise set to `false`
callTo1	set to `true` if the elevator is called to floor 1; otherwise set to `false`
callTo2	set to `true` if the elevator is called to floor 2; otherwise set to `false`

The elevator moves when the door is closed and the elevator is called to the floor that it is not currently on. Which of the following Boolean expressions can be used in a selection statement to cause the elevator to move?

(A) (onFloor1 AND callTo2) AND (onFloor2 AND callTo1)

(B) (onFloor1 AND callTo2) OR (onFloor2 AND callTo1)

(C) (onFloor1 OR callTo2) AND (onFloor2 OR callTo1)

(D) (onFloor1 OR callTo2) OR (onFloor2 OR callTo1)

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
5.5 Programming uses mathematical and logical concepts.	**5.5.1** Employ appropriate mathematical and logical concepts in programming. [P4]	**P1** Connecting computing	**5.5.1E** **5.5.1F** **5.5.1G**

20. According to the domain name system (DNS), which of the following is a subdomain of the domain `example.com`?

(A) `about.example.com`

(B) `example.co.uk`

(C) `example.com.org`

(D) `example.org`

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
6.2 Characteristics of the Internet influence the systems built on it.	**6.2.1** Explain characteristics of the Internet and the systems built on it. [P5]	**P5** Communicating	**6.2.1B**

21. Which of the following algorithms require <u>both</u> selection and iteration?

    Select <u>two</u> answers.

    (A) An algorithm that, given two integers, displays the greater of the two integers

    (B) An algorithm that, given a list of integers, displays the number of even integers in the list

    (C) An algorithm that, given a list of integers, displays only the negative integers in the list

    (D) An algorithm that, given a list of integers, displays the sum of the integers in the list

Enduring Understandings	Learning Objectives	Computational Thinking Practices	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.	**4.1.1** Develop an algorithm for implementation in a program. [P2]	**P2** Creating computational artifacts	**4.1.1A** **4.1.1C** **4.1.1D**

22. A teacher uses the following program to adjust student grades on an assignment by adding 5 points to each student's original grade. However, if adding 5 points to a student's original grade causes the grade to exceed 100 points, the student will receive the maximum possible score of 100 points. The students' original grades are stored in the list `gradeList`, which is indexed from 1 to `n`.

```
i ← 1

REPEAT n TIMES
{
    <MISSING CODE>
    i ← i + 1
}
```

The teacher has the following procedures available.

Procedure	Explanation
`min (a, b)`	Returns the lesser of the two values a and b
`max (a, b)`	Returns the greater of the two values a and b

Which of the following code segments can replace `<MISSING CODE>` so that the program works as intended?

Select <u>two</u> answers.

(A) `gradeList[i] ← min (gradeList[i] + 5, 100)`

(B) `gradeList[i] ← max (gradeList[i] + 5, 100)`

(C)
```
gradeList[i] ← gradeList[i] + 5
IF (gradeList[i] > 100)
{
gradeList[i] ← gradeList[i] - 5
}
```

(D)
```
gradeList[i] ← gradeList[i] + 5
IF (gradeList[i] > 100)
{
gradeList[i] ← 100
}
```

Enduring Understandings	Learning Objectives	Computational Thinking Practices	Essential Knowledge
5.5 Programming uses mathematical and logical concepts.	**5.5.1** Employ appropriate mathematical and logical concepts in programming. [P1]	**P1** Connecting computing	**5.5.1A** **5.5.1B** **5.5.1H**

Answers to Sample Exam Questions

1 – C	12 – A
2 – C	13 – C
3 – C	14 – C
4 – A	15 – C
5 – A	16 – A
6 – C	17 – D
7 – C	18 – D
8 – D	19 – B
9 – D	20 – A
10 – B	21 – B, C
11 – C	22 – A, D

Reproducibles for Students

The following pages contain reproducible versions of the assessment overview and performance task guidelines.

AP Computer Science Principles Assessment Overview for Students

The AP Computer Science Principles course has three assessments, consisting of two performance tasks and an end-of-course AP Exam. All three assessments are summative and will be used to calculate a final AP score (using the 1–5 scale) for AP Computer Science Principles.

Assessment	Timing	Percentage of Total AP Score
Explore Performance Task	8 hours	16%
Create Performance Task	12 hours	24%
End-of-Course Exam	2 hours	60%

Students who are completing the AP Computer Science Principles course in a nontraditional classroom situation (e.g., online, homeschool, independent study) should consult a school-based AP Coordinator for instructions on taking the AP Exam and submitting work for the performance tasks.

Investigation and Citation

The through-course performance tasks require you to create computational artifacts. A computational artifact is a visualization, a graphic, a video, a program, or an audio recording that you create using a computer. For the Create performance task, you will develop a computer program and for the Explore performance task, you will create a computational artifact of your choosing to represent or illustrate the intended purpose, function, or effect of a computing innovation using any computational tool(s) you wish.

In creating your computational artifact, you can create your own original work, including video, music, text, images, graphs, and program code. If you use external work to integrate into your computational artifact, you must acknowledge, attribute, and/or cite sources and include a bibliography with your submission. External work that should be acknowledged include video, music, text, images, graphs, and program code that are used in the creation of your computational artifacts.

AP Computer Science Principles Policy on Plagiarism

A student who fails to acknowledge (i.e., through citation, through attribution, by reference, and/or through acknowledgment in a bibliographic entry) the source or author of any and all information or evidence taken from the work of someone else will receive a score of 0 on that performance task.

To the best of their ability, teachers will ensure that students understand ethical use and acknowledgment of the ideas and work of others as well as the consequences of plagiarism. The student's individual voice should be clearly evident, and the ideas of others must be acknowledged, attributed, and/or cited. A computational artifact without acknowledgment of the media used in the creation of the computational artifact, and program code segment(s) written by someone else used in a program without appropriate acknowledgment, are all considered plagiarized work.

Programming Language Requirements

AP Computer Science Principles is language agnostic. This means that there is no specific language requirement. When completing the Create – Applications from Ideas performance task for this course, you are allowed to select a language you feel is most appropriate to meet the requirements of the task. When selecting a language or program, you should review the requirements section of the performance task to ensure that your program will be sophisticated enough to implement mathematical and logical concepts, create abstractions, and implement algorithms.

Peer-to-Peer Collaboration

Collaboration is only allowed on designated sub-components of the Create performance task.

For the Explore – Impact of Computing Innovations performance task, collaboration of any kind is not allowed.

For the Create – Applications from Ideas performance task, you are encouraged to collaborate on the development of their program with another student in your class. Collaboration is not allowed during the creation of the video or when answering the written responses.

Students completing AP Computer Science Principles in a nontraditional classroom situation (e.g., online, homeschool, independent study) are encouraged to collaborate with another student peer when completing the Create performance task.

Preparing for the Through-Course Performance Tasks

The following guidelines are meant to help you be successful on the performance tasks as well as to clarify or address any questions you may have regarding the process of completing these tasks.

Prior to your teacher administering the performance tasks, you should:

- obtain content knowledge and skills that will help you succeed on the performance tasks;

- practice either an entire performance task or individual prompts of the tasks;

- review the scoring guidelines, found on AP Central, to understand how your work will be assessed;

- examine examples of performance task found on AP Central of submissions at high, medium, and low levels. If you select a computing innovation that is represented in one of the examples, or that was discussed in class, you must find new sources and submit original responses to avoid. You cannot submit any work from practice performance tasks.

- pay attention to the instructions concerning the size of the files to be uploaded; the computational artifact for the Explore performance task and the video for the Create performance task individually cannot exceed 30MB.

- ensure you know the proper way to evaluate and appropriately cite a source, including program code; any program code which has not been written by you must be cited and credit given to the author;

- understand the level of detail expected in writing your responses by examining the scoring guidelines and high level samples found on AP Central;

- understand that you may not revise your work once you have submitted your work as final to the AP Digital Portfolio; and

- be aware that the scoring process that occurs in the AP Reading may be different from the scoring process that occurs in your classroom; the AP score that you receive may be different than your classroom grade.

Performance Task: Explore – Impact of Computing Innovations

Overview

Computing innovations impact our lives in ways that require considerable study and reflection for us to fully understand them. In this performance task, you will explore a computing innovation of your choice. A computing innovation is an innovation that includes a computer or program code as an integral part of its functionality. Your close examination of this computing innovation will deepen your understanding of computer science principles.

Please note that once this performance task has been assigned as an assessment (rather than as practice), you are expected to complete the task with minimal assistance from anyone. For more clarification see the Guidelines for Completing the Through-Course Performance Tasks section.

You will be provided with a minimum of 8 hours of class time to develop, complete, and submit the following:

▶ **A computational artifact**

▶ **Written responses**

Scoring guidelines and instructions for submitting your performance tasks are available on the AP Computer Science Principles Course Home Page.

Note: Students in nontraditional classroom environments should consult a school-based AP Coordinator for submission instructions.

When completing the Explore – Impacts of Computing Innovations performance task, you will be expected to conduct investigations on a computing innovation. A computing innovation is an innovation that includes a computer or program code as an integral part of its functionality.

You must ensure you have identified relevant, credible, and easily accessible sources to support your creation of a computational artifact as well as to support your responses to the prompts. You can search for print or nonprint sources as part of your investigation. You can refer to a journal, Web page, or an expert that is being quoted as part of your written response. Avoid plagiarism by acknowledging, attributing, and/or citing sources throughout your responses.

General Requirements

This performance task requires you to select and investigate a computational innovation to:

▶ analyze a computing innovations impact on society, economy, or culture and explain how this impact could be beneficial and/or harmful;

▸ explain how a computing innovation consumes, produces, or transforms data; and

▸ describe how data storage, data privacy, or data security concerns are raised based on the capabilities of the computing innovation.

You are also required to:

▸ investigate your computing innovation using a variety of sources (e.g., print, online, expert interviews);

▸ provide in-text citations of at least three different sources that helped you create your computational artifact and/or formulate your written responses;

› At least two of the sources must be available online or in print; your third source may be either online, in print, or a personal interview with an expert on the computing innovation.

› At least two of the sources must have been created after the end of the previous academic year.

▸ produce a computational artifact that illustrates, represents, or explains the computing innovation's intended purpose, its function, or its effect; and

▸ provide written responses to all the prompts in the performance task about your computational artifact and computing innovation.

Submission Requirements

1. **Computational Artifact**

   Your computational artifact must provide an illustration, representation, or explanation of the computing innovation's intended purpose, its function, or its effect. The computational artifact must not simply repeat the information supplied in the written responses and should be primarily nontextual.

   Submit a video, audio, or PDF file. Use computing tools and techniques to create one original computational artifact (a visualization, a graphic, a video, a program, or an audio recording). **Acceptable multimedia file types include .mp3, .mp4, .wmv, .avi, .mov, .wav, .aif, or .pdf format. PDF files must not exceed three pages. Video or audio files must not exceed 1 minute in length and must not exceed 30MB in size.**

2. **Written Responses**

   Submit one PDF file in which you respond directly to each of the prompts below. **Clearly label your responses 2a–2e in order.** Your responses must provide evidence of the extensive knowledge you have developed about your chosen computing innovation and its impact(s). Write your responses so they would be understandable to someone who is not familiar with the computing innovation. Include citations, as applicable, within your written responses. **Your response to prompts 2a–2d combined must not exceed 700 words.** The references required in 2e are not included in the final word count.

Computational Artifact

2a. Provide information on your computing innovation and computational artifact.

- Name the computing innovation that is represented by your computational artifact.

- Describe the computing innovation's intended purpose and function.

- Describe how your computational artifact illustrates, represents, or explains the computing innovation's intended purpose, its function, or its effect.

(*Must not exceed 100 words*)

2b. Describe your development process, explicitly identifying the computing tools and techniques you used to create your artifact. Your description must be detailed enough so that a person unfamiliar with those tools and techniques will understand your process. (*Must not exceed 100 words*)

Computing Innovation

2c. Explain at least one beneficial effect and at least one harmful effect the computing innovation has had, or has the potential to have, on society, economy, or culture. (*Must not exceed 250 words*)

2d. Using specific details, describe:

- the data your innovation uses;

- how the innovation consumes (as input), produces (as output), and/or transforms data; and

- at least one data storage concern, data privacy concern, or data security concern directly related to the computing innovation.

(*Must not exceed 250 words*)

References

2e. Provide a list of at least three online or print sources used to create your computational artifact and/or support your responses through in-text citation to the prompts provided in this performance task.

- At least two of the sources must have been created after the end of the previous academic year.

- For each online source, include the complete and permanent URL. Identify the author, title, source, the date you retrieved the source, and, if possible, the date the reference was written or posted.

- For each print source, include the author, title of excerpt/article and magazine or book, page number(s), publisher, and date of publication.

- If you include an interview source, include the name of the person you interviewed, the date on which the interview occurred, and the person's position in the field.

- Include in-text citations for the sources you used.

- Each source must be relevant, credible, and easily accessed.

Preparing for the Explore Performance Task

Prior to your teacher administering this task, you should:

▶ understand that a computing innovation (i.e., an innovation that that includes a computer or program code as an integral part of its functionality) has a meaningful personal or community emphasis is an appropriate choice, as long as it fulfills the requirements to complete all the prompts in the performance task;

▶ practice searching and evaluating sources relevant to computing innovations; all sources cited must be relevant, credible, and easily accessible;

▶ practice clearly explaining the impact the intended use of a computing innovation has on society, economy, and culture, clearly justifying both beneficial and harmful effects;

▶ practice demonstrating your knowledge of computer science and understanding of how data is input, output, and transformed in your analysis of the data used by the computing innovation.

▶ practice making connections between the data used by a computing innovation and a security, privacy, or storage concern.

▶ obtain the meaning and purpose of creating a computational artifact; your creation must provide an illustration, representation, or explanation of the computing innovation's intended purpose, its function, or its effect;

▶ have exposure to the use of a variety of computational tools that can be used to create effective computational artifacts;

▶ understand which computational artifacts would be considered effective and ineffective.

   Effective artifacts include:

   › visual, graphical, and/or audio content to help a reader understand the purpose, function, or effect of a computing innovation; and

   › the use of communications media, such as animations, comic strips, infographics, and/or public service announcements, to illustrate the purpose, function, or effect of a computing innovation.

   Ineffective artifacts include:

   › artifacts that repeat information previously supplied in the written responses;

   › multislide presentations with paragraphs of text or bullets;

   › artifacts that have not been created by the student; and

   › artifacts that focus on advertising the computing innovation's functionality instead of the purpose of the innovation.

▶ practice writing responses based on relevant and credible sources and include in-text citations; and

▶ practice appropriate acknowledgment of sources used in the creation of your computational artifact.

Guidelines for Completing the Explore Performance Task

You must:

▸ be aware of the performance task directions, timeline, and scoring criteria;

▸ support your written analysis of your computing innovation when responding to all the prompts by using details related to the knowledge and understanding of computer science you have obtained throughout the course and your investigation;

▸ provide evidence to support your claims using in-text citations;

▸ use relevant and credible sources to gather information about your computing innovation;

▸ provide acknowledgments for the use of any media or program code used in the creation of your computational artifact that is not your own; and

▸ allow your own interests to drive your choice of computing innovation and computational artifact.

You may:

▸ follow a timeline and schedule for completing the performance task;

▸ seek clarification from your teacher or AP Coordinator pertaining to the task, timeline, components, and scoring criteria;

▸ seek clarification from your teacher or AP Coordinator regarding submission requirements;

▸ as needed, seek assistance from your teacher or AP Coordinator in defining your focus and choice of topics; and

▸ seek assistance from your teacher or AP Coordinator to resolve technical problems that impede work, such as a failing workstation or difficulty with access to networks, or help with saving files or making movie files.

You may not:

▸ collaborate on the Explore performance task;

▸ submit work that has been revised, amended, or corrected by another individual;

▸ submit work from a practice performance task as your official submission to the College Board to be scored by the AP Program; or

▸ seek assistance or feedback on answers to prompts.

Performance Task: Create – Applications from Ideas

Overview

Programming is a collaborative and creative process that brings ideas to life through the development of software. Programs can help solve problems, enable innovations, or express personal interests. In this performance task, you will be developing a program of your choice. Your development process should include iteratively designing, implementing, and testing your program. You are strongly encouraged to work with another student in your class.

Please note that once this performance task has been assigned as an assessment (rather than as practice), you are expected to complete the task with minimal assistance from anyone other than your collaborative peer(s). For more clarification see the Guidelines for Completing the Through-Course Performance Tasks section.

You will be provided with a minimum of 12 hours of class time to complete and submit the following:

▸ **A video of your program running**

▸ **Individual written responses about your program and development process**

▸ **Program code**

Scoring guidelines and instructions for submitting your performance tasks are available on the AP Computer Science Principles Course Home Page.

Note: Students in nontraditional classroom environments should consult a school-based AP Coordinator for instructions.

General Requirements

This performance task requires you to develop a program on a topic that interests you or one that solves a problem. During the completion of this performance task, you will iteratively design, implement, and test your program. You will provide written responses to prompts about your program and specific program code that are significant to the functionality of your program. It is strongly recommended that a portion of the program involve some form of collaboration with another student in your class, for example, in the planning, designing, or testing (debugging) part of the development process. Your program development must also involve a significant amount of independent work writing your program code, in particular, algorithm(s) and abstraction(s) that you select to use as part of your written response to describe how the program code segments help your program run.

You are required to:

▸ independently develop an algorithm that integrates two or more algorithms and that is fundamental for your program to achieve its intended purpose;

▸ develop an abstraction that manages the complexity of your program;

▸ create a video that displays the running of your program and demonstrates its functionality;

▸ write responses to all the prompts in the performance task; and

▸ submit your entire program code.

Program Requirements

Your program must demonstrate a variety of capabilities and implement several different language features that, when combined, produce a result that cannot be easily accomplished without computing tools and techniques. Your program should draw upon mathematical and logical concepts, such as use of numbers, variables, mathematical expressions with arithmetic operators, logical and Boolean operators and expressions, decision statements, iteration, and/or collections.

Your program must demonstrate:

▸ use of several effectively integrated mathematical and logical concepts, from the language you are using;

▸ implementation of an algorithm that integrates two or more algorithms and integrates mathematical and/or logical concepts; and

▸ development and use of abstractions to manage the complexity of your program (e.g., procedures, abstractions provided by the programming language, APIs).

Submission Requirements

1. **Video**

   Submit one video in .mp4, .wmv, .avi, or .mov format that demonstrates the running of at least one significant feature of your program. **Your video must not exceed 1 minute in length and must not exceed 30MB in size.**

2. **Written Responses**

   Submit one PDF file in which you respond directly to each prompt. **Clearly label your responses 2a–2d in order. Your response to all prompts combined must not exceed 750 words, exclusive of the Program Code.**

Program Purpose and Development

2a. Provide a written response or audio narration in your video that:

- identifies the programming language;

- identifies the purpose of your program; and

- explains what the video illustrates.

(Must not exceed 150 words)

2b. Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and/ or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development. *(Must not exceed 200 words)*

2c. Capture and paste a program code segment that implements an algorithm (marked with an **oval** in **section 3** below) and that is fundamental for your program to achieve its intended purpose. This code segment must be an algorithm you developed individually on your own, must include two or more algorithms, and must integrate mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program. *(Must not exceed 200 words)*

2d. Capture and paste a program code segment that contains an abstraction you developed individually on your own (marked with a **rectangle** in **section 3** below). This abstraction must integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. *(Must not exceed 200 words)*

3. **Program Code**

Capture and paste your entire program code in this section.

› Mark with an **oval** the segment of program code that implements the algorithm you created for your program that integrates other algorithms and integrates mathematical and/or logical concepts.

› Mark with a **rectangle** the segment of program code that represents an abstraction you developed.

› Include comments or acknowledgments for program code that has been written by someone else.

Preparing for the Create Performance Task

Prior to your teacher administering this task, you should:

▸ Brainstorm problems that programming can address, or brainstorm special interests that programming can help develop;

▸ Ensure you understand the iterative nature of developing a computer program;

▸ Be prepared to collaborate with peers as necessary and in different ways;

▸ Ensure you are able to analyze program code and code segments and explain the function as it relates to the overall program;

▸ Know how to keep a programming journal of the design choices that you will make during the development of your program code and the effect of these decisions on the program's function. You can use this journal as a point of reference when demonstrating your understanding of how:

> › an algorithm was built as part of the integration of two or more algorithms;

> › a program functions differently with the inclusion of algorithms and abstractions;

> › the inclusion of an abstraction has made their program code more compact, readable and/or reusable and how the program would operate differently without the inclusion of the abstraction;

▸ obtain programming support as necessary while practicing the skills needed to complete the performance task.

Guidelines for Completing the Create Performance Task

You must:

▸ be aware of the performance task directions, timeline, and scoring criteria;

▸ apply computer science knowledge you've obtained throughout the course when developing your program and in your explanation of its function;

▸ provide acknowledgment for program code used in your program that is not your own; and

▸ allow your own interests to drive your choice of program.

You may:

▸ follow a timeline and schedule for completing the performance task;

▸ seek clarification from your teacher or AP Coordinator pertaining to the task;

▸ seek clarification from your teacher or AP Coordinator regarding submission requirements;

- as needed, seek assistance from your teacher or AP Coordinator in defining your focus or choice of topics;

- seek assistance from your teacher or AP Coordinator to resolve technical problems that impede work, such as a failing workstation or difficulty with access to networks, or help with saving files or making movie files;

- obtain assistance from your teacher or AP Coordinator with the formation of peer-to-peer collaboration when completing the Create performance task;

- seek assistance from your teacher or AP Coordinator in resolving collaboration issues where one partner is clearly and directly impeding the completion of the Create performance task; and

- seek guidance from your teacher or AP Coordinator to use and acknowledge APIs or other pieces of open-source code. Program code not written by you can be used in programs as long as you are extending the project in some new way. You should provide acknowledgment and credit from program code you did not write.

You may not:

- collaborate on the video or any of the written responses;

- submit work that has been revised, amended, or correct by another individual, with the exception of cited program code;

- submit work from a practice performance task as your official submission to the College Board to be scored by the AP Program; or

- seek assistance or feedback on answers to prompts.

AP Computer Science Principles Exam Reference Sheet
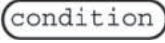
As AP Computer Science Principles does not designate any particular programming language, this reference sheet provides instructions and explanations to help students understand the format and meaning of the questions they will see on the exam. The reference sheet includes two programming formats: text based and block based.
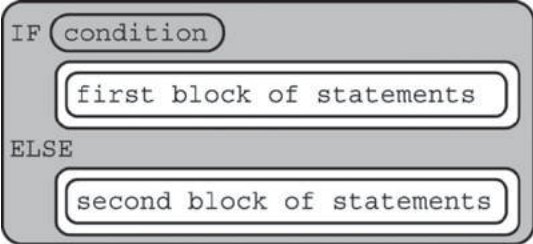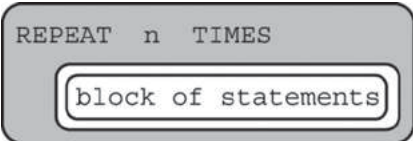
Programming instructions use four data types: numbers, Booleans, strings, and lists.

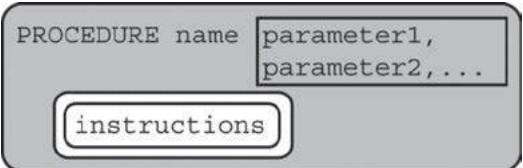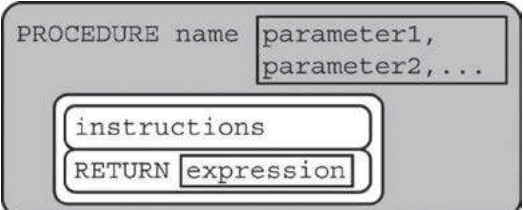Instructions from any of the following categories may appear on the exam:

▶ Assignment, Display, and Input

▶ Arithmetic Operators and Numeric Procedures

▶ Relational and Boolean Operators

▶ Selection

▶ Iteration

▶ List Operations

▶ Procedures

▶ Robot

Instruction	Explanation
Assignment, Display, and Input	

Text:
`a ← expression`
 	
Block:
`a ⟵ expression`
	Evaluates `expression` and assigns the result to the variable `a`.

Text:
`DISPLAY (expression)`
 	
Block:
`DISPLAY expression`
	Displays the value of `expression`, followed by a space.

Text:
`INPUT ()`
 	
Block:
`INPUT`
	Accepts a value from the user and returns it.

Arithmetic Operators and Numeric Procedures	

Text and Block:
`a + b`
`a - b`
`a * b`
`a / b`
	The arithmetic operators `+`, `-`, `*`, and `/` are used to perform arithmetic on `a` and `b`.
	
	For example, `3 / 2` evaluates to `1.5`.

Text and Block: `a MOD b`	Evaluates to the remainder when `a` is divided by `b`. Assume that `a` and `b` are positive integers. For example, `17 MOD 5` evaluates to `2`.

Text:
`RANDOM (a, b)`
 	
Block:
`RANDOM a, b`
	Evaluates to a random integer from `a` to `b`, including `a` and `b`.
	
	For example, `RANDOM (1, 3)` could evaluate to `1`, `2`, or `3`.

Relational and Boolean Operators	

Text and Block:
`a = b`
`a ≠ b`
`a > b`
`a < b`
`a ≥ b`
`a ≤ b`
	The relational operators =, ≠, >, <, ≥, and ≤ are used to test the relationship between two variables, expressions, or values.
	
	For example, `a = b` evaluates to `true` if `a` and `b` are equal; otherwise, it evaluates to `false`.

Instruction	Explanation
Relational and Boolean Operators (continued)	

Text:
`NOT condition`
 	
Block:
`NOT (condition)`
	Evaluates to `true` if condition is `false`; otherwise evaluates to `false`.

Text:
`condition1 AND condition2`
 	
Block:
`(condition1) AND (condition2)`
	Evaluates to `true` if both `condition1` and `condition2` are `true`; otherwise, evaluates to `false`.

Text:
`condition1 OR condition2`
 	
Block:
`(condition1) OR (condition2)`
	Evaluates to `true` if `condition1` is `true` or if `condition2` is `true` or if both `condition1` and `condition2` are `true`; otherwise, evaluates to `false`.

Selection	

Text:
`IF (condition)`
`{`
`   <block of statements>`
`}`
 	
Block:
`IF (condition)`
`   block of statements`
	The code in `block of statements` is executed if the Boolean expression `condition` evaluates to `true`; no action is taken if `condition` evaluates to `false`.

Instruction	Explanation
Selection (continued)	

Text:
``IF (condition)
{
    <first block of statements>
}
ELSE
{
    <second block of statements>
}```
 	
Block:
 	
	The code in `first block of statements` is executed if the Boolean expression `condition` evaluates to `true`; otherwise, the code in `second block of statements` is executed.

Iteration	

Text:
``REPEAT n TIMES
{
    <block of statements>
}```
 	
Block:
 	
	The code in `block of statements` is executed `n` times.

Text:
``REPEAT UNTIL (condition)
{
    <block of statements>
}```
 	
Block:
 	
	The code in `block of statements` is repeated until the Boolean expression `condition` evaluates to `true`.

Instruction	Explanation
List Operations	
For all list operations, if a list index is less than 1 or greater than the length of the list, an error message is produced and the program terminates.	

Text:
`list[i]`
 	
Block:
`list i`
	Refers to the element of `list` at index `i`. The first element of `list` is at index `1`.

Text:
`list[i] ← list[j]`
 	
Block:
`list i ← list j`
	Assigns the value of `list[j]` to `list[i]`.

Text:
`list ← [value1, value2, value3]`
 	
Block:
`list ← value1, value2, value3`
	Assigns `value1`, `value2`, and `value3` to `list[1]`, `list[2]`, and `list[3]`, respectively.

Text:
`FOR EACH item IN list`
`{`
`    <block of statements>`
`}`
 	
Block:
`FOR EACH item IN list`
`   block of statements`
	The variable `item` is assigned the value of each element of `list` sequentially, in order from the first element to the last element. The code in `block of statements` is executed once for each assignment of `item`.

Text:
`INSERT (list, i, value)`
 	
Block:
`INSERT list, i, value`
	Any values in `list` at indices greater than or equal to `i` are shifted to the right. The length of list is increased by 1, and `value` is placed at index `i` in `list`.

Text:
`APPEND (list, value)`
 	
Block:
`APPEND list, value`
	The length of `list` is increased by 1, and `value` is placed at the end of `list`.

Instruction	Explanation
List Operations (continued)	

Text:
`REMOVE (list, i)`
Block:
`REMOVE list, i`
	Removes the item at index `i` in `list` and shifts to the left any values at indices greater than `i`. The length of `list` is decreased by 1.

Text:
`LENGTH (list)`
Block:
`LENGTH list`
	Evaluates to the number of elements in `list`.

Procedures	

Text:
`PROCEDURE name (parameter1,`
`              parameter2, ...)`
`{`
`    <instructions>`
`}`
Block:
`PROCEDURE name parameter1, parameter2,...`
`instructions`
	A procedure, `name`, takes zero or more parameters. The procedure contains programming instructions.

Text:
`PROCEDURE name (parameter1,`
`              parameter2, ...)`
`{`
`    <instructions>`
`    RETURN (expression)`
`}`
Block:
`PROCEDURE name parameter1, parameter2,...`
`instructions`
`RETURN expression`
	A procedure, `name`, takes zero or more parameters. The procedure contains programming instructions and returns the value of `expression`. The `RETURN` statement may appear at any point inside the procedure and causes an immediate return from the procedure back to the calling program.

Instruction	Explanation
Robot	
If the robot attempts to move to a square that is not open or is beyond the edge of the grid, the robot will stay in its current location and the program will terminate.	
Text: MOVE_FORWARD () Block: `MOVE_FORWARD`	The robot moves one square forward in the direction it is facing.
Text: ROTATE_LEFT () Block: `ROTATE_LEFT`	The robot rotates in place 90 degrees counterclockwise (i.e., makes an in-place left turn).
Text: ROTATE_RIGHT () Block: `ROTATE_RIGHT`	The robot rotates in place 90 degrees clockwise (i.e., makes an in-place right turn).
Text: CAN_MOVE (direction) Block: `CAN_MOVE` `direction`	Evaluates to `true` if there is an open square one square in the direction relative to where the robot is facing; otherwise evaluates to `false`. The value of direction can be `left`, `right`, `forward`, or `backward`.

Appendix: Changes to the Course and Exam Description

Since the publication's update in fall 2016 some modifications have been made to the AP Computer Science Principles curriculum framework. The chart below summarizes the changes made, which are now reflected in this course and exam description.

Revision	Explanation
1. Definition of computing innovation	A computing innovation is an innovation that includes a computer or program code as an integral part of its functionality.
2. Guidance to build collaboration throughout the course	This information has been revised and moved from the assessment section into the instructional approaches section to provide guidance on building collaboration throughout the course.
3. Clarification on in-text citation	Students must support their claims with at least three in-text citations within the written responses submitted for the Explore performance task.
4. Guidance on the development and selection of algorithms and abstractions.	The algorithms (for prompt 2c) and the abstraction (for prompt 2d) selected as part of the written responses for the Create performance task must be developed independently, not collaboratively with a peer(s).
5. Organization of performance tasks guidelines	The following changes were made to the performance tasks guidelines: ▶ Teacher guidelines (i.e., what teachers "should" do to prepare their students, and what teachers "must", "may", and "may not" do during the administration of the performance tasks) have been moved to appear after the task guidelines for each performance task. ▶ Additional guidelines have been added for clarification.
6. Reproducibles for Students section	The following changes were made to the Reproducibles for Students section: ▶ Guidelines for preparing for the performance tasks (i.e., what students "must", "may", and "may not" do during the administration of each performance task) have been moved to appear after the task directions for each performance task. ▶ Additional guidelines have been added for clarification and to mirror the guidelines provided to teachers.
7. Sample Questions	All sample questions have been aligned to only one learning objective.

APPENDIX

Contact Us

AP Services
P.O. Box 6671
Princeton, NJ 08541-6671
609-771-7300
888-225-5427 (toll free in the
U.S. and Canada)
610-290-8979 (fax)
Email: apexams@info.collegeboard.org

National Office
250 Vesey Street
New York, NY 10281
212-713-8000

AP Canada Office
2950 Douglas Street, Suite 550
Victoria, BC, Canada V8T 4N4
250-472-8561
800-667-4548 (toll free in Canada only)
Email: gewonus@ap.ca

International Services
Serving all countries outside the U.S. and Canada
250 Vesey Street
New York, NY 10281
212-373-8738
Email: international@collegeboard.org

Middle States Regional Office
*Serving Delaware, District of Columbia, Maryland,
New Jersey, New York, Pennsylvania, Puerto Rico,
and the U.S. Virgin Islands*
Three Bala Plaza East
Suite 501
Bala Cynwyd, PA 19004-1501
610-227-2550
866-392-3019
610-227-2580 (fax)
Email: msro@info.collegeboard.org

Midwestern Regional Office
*Serving Illinois, Indiana, Iowa, Kansas, Michigan,
Minnesota, Missouri, Nebraska, North Dakota,
Ohio, South Dakota, West Virginia, and Wisconsin*
6111 N. River Road, Suite 550
Rosemont, IL 60018-5158
866-392-4086
847-653-4528 (fax)
Email: mro@info.collegeboard.org

New England Regional Office
*Serving Connecticut, Maine, Massachusetts, New
Hampshire, Rhode Island, and Vermont*
1601 Trapelo Road, Suite 12
Waltham, MA 02451-1982
866-392-4089
781-663-2743 (fax)
Email: nero@info.collegeboard.org

Southern Regional Office
*Serving Alabama, Florida, Georgia, Kentucky,
Louisiana, Mississippi, North Carolina, South
Carolina, Tennessee, and Virginia*
3700 Crestwood Parkway NW, Suite 700
Duluth, GA 30096-7155
866-392-4088
770-225-4062 (fax)
Email: sro@info.collegeboard.org

Southwestern Regional Office
Serving Arkansas, New Mexico, Oklahoma, and Texas
4330 Gaines Ranch Loop, Suite 200
Austin, TX 78735-6735
866-392-3017
512-721-1841 (fax)
Email: swro@info.collegeboard.org

Western Regional Office
*Serving Alaska, Arizona, California, Colorado,
Hawaii, Idaho, Montana, Nevada, Oregon, Utah,
Washington, and Wyoming*
2099 Gateway Place, Suite 550
San Jose, CA 95110-1051
866-392-4078
408-367-1459 (fax)
Email: wro@info.collegeboard.org

apcentral.collegeboard.org

00677-014
160081394