The College Board

AP® Computer Science Principles
Draft Curriculum Framework

February 2014

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

Introduction

AP® Computer Science Principles is designed to introduce students to the central ideas of computer science, to instill ideas and practices of computational thinking, and to have students engage in activities that show how computing changes the world. The course is rigorous and rich in computational content, includes computational and critical thinking skills, and engages students in the creative aspects of the field. Through both its content and pedagogy, this course aims to appeal to a broad audience.

This intellectually rich and engaging course emphasizes three key themes that help students build a solid understanding and facility with computing and computational thinking — understandings that are important, if not integral, to being part of a well-educated and informed citizenry.

The first theme of the AP Computer Science Principles course is its focus on creativity. The computational thinking practices and big ideas that follow hint at the creative nature of computing, yet alone they cannot truly convey the importance of creativity in this course. It's not enough for students to know that "computing requires creativity." Rather, students must actually *be creative*: creating artifacts that they want to show off to their friends and family, using simulation to explore questions that interest them, and designing and implementing solutions employing the iterative and sometimes messy process that artists, writers, computer scientists, and engineers use to translate ideas into tangible form.

A second theme is the course's use of technology as a means for solving computational problems and exploring creative endeavors, rather than a focus on a specific tool or programming language. To that end, the course highlights programming as one of the seven big ideas of computer science, because programming is among the creative processes that help transform ideas into reality. Programming is a tool students use to explore concepts and create exciting and personally relevant artifacts. In contrast to traditional college introductory computer science courses and the AP Computer Science A course, the AP Principles course does not focus on and is not organized around a specific language. The instructor of the course selects one or more languages, based on appropriateness for a specific project or problem and according to guidelines provided as part of the course specification. Language specifics are taught only to the extent that students need them to produce their programs. Similarly, data, and the use of computational tools to analyze and study data, is another of the big ideas of computer science, as data plays an incredibly important role in so many aspects of our lives. Students in this course work with large data sets — they analyze, visualize, draw conclusions from trends— but the course itself does not specify particular computing tools or the use of specific programming languages for these explorations.

A third theme that helps the course appeal to a broad audience is the course's focus on people and society, not just on machines and systems. Students in an AP Computer Science Principles course explore computer science's relevance to and impact on the world today. They investigate the innovations in other fields that computing has made possible. They examine the ethical implications of new computing technologies. They perform activities that develop their communication and collaboration skills. Students in this course work individually and collaboratively to solve problems. They talk and write about their solutions, the importance of these problems, and their impact on the world.

This curriculum framework specifies the course curriculum: the content, practices, thinking, and skills central to the discipline of computer science. Through this novel content with implications for engaging

pedagogy, students will experience the joy and beauty that permeates computing: They will not only experience the sense of community from connecting with friends on social networks, but they will understand many aspects of the software and algorithms that make these social networks possible. They will not only use algorithms, but also create them and experience the "ah ha!" moment when an algorithm finally makes sense. They will not simply run programs; they will experience the thrill of constructing a program and seeing it work, as well as the pride of creating something for oneself, one's family or friends, or for the world.

Overview of the Curriculum Framework

This curriculum framework is designed to provide a clear and detailed description of the course curriculum and course content. The key sections of this framework are described below.

- The **computational thinking practices skills** capture important aspects of the work that computer scientists engage in at the level of competence expected of AP Computer Science Principles students. The practices and skills help students coordinate and make sense of knowledge in order to accomplish a goal or task. They enable students to engage with the AP CSP course content by developing computational artifacts and analyzing data, information, or knowledge represented for computational use. In addition, learning to collaborate to build computational artifacts and to communicate their purpose is a requirement for students to be successful in this course. Because content knowledge and practices and skills are equally important in Computer Science Principles, each learning objective includes a direct correlation to a computational thinking practice or skill.

- The key concepts and related content that define the Principles course and exam are organized around seven **big ideas**, which encompass fundamental ideas foundational to computing. These big ideas connect students to a curriculum scope that includes the art of programming but is not programming focused. For each of the big ideas, **enduring understandings**, which incorporate the core concepts that students should retain from their learning experiences, are also identified.

- Each enduring understanding is followed by at least one or more **learning objective** that provides clear and detailed **articulation** of what students should know and be able to do. The learning objectives are designed to help teachers integrate the computational thinking practices with specific content, and to provide them clear information about how students will be expected to demonstrate their knowledge and abilities. They are numbered to correspond with the big ideas and enduring understandings (e.g., LO 1.1.1).

- Next to each learning objective is a listing of **essential knowledge statements. These statements specify facts** or content that students must know in order to be able successfully perform the learning objectives.. These additional underlying content components are listed numerically in the column next to the supported learning objective, and each one includes one or more bulleted statements describing further content details. All examples and content references are considered to be required and may be the focus of exam questions. For example, the following essential knowledge content statements correspond to Objective 1.1.1 Apply a creative development process when creating computational artifacts. [P2]:

1.1.1A A creative process in the development of a computational artifact can include but is not limited to employing non-traditional, non-prescribed 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.

Relationship between the Curriculum Framework and the Assessments

The learning objectives will be targets of two different types of assessment: an end-of-course AP Exam and a through-course AP assessment component. The AP Computer Science Principles Exam will be a computer-based exam. The exam will contain a variety of question types, including single and multiple select, drag and drop, and limited written response within a text box. Computer specifications for the AP Computer Science Principles Exam can be found on the AP Computer Science Principles Course Home Page.

The through-course component is comprised of three performance tasks — separately, these tasks require students to conduct investigations using data, explore impacts of computing, and create computational artifacts through programming.

Students will be asked to demonstrate understanding by applying computer science skills and practices to the learning objectives, including related content from the essential knowledge statements.

Like an exam, the performance tasks are designed to gather *evidence* of student learning with regard to the learning objectives. The tasks measure identified learning objectives, which include computational thinking practices.

Performance tasks assess student achievement in more robust ways than are available on a timed exam. Additionally, there are a number of learning objectives that are difficult to measure using a traditional exam but that lend themselves well to a performance task.

These performance tasks require an extended level of effort. Depending on their nature, they could take several weeks to complete. Each contains specific requirements and a list of learning objectives addressed by the task. For more information about the AP Computer Science Principles Performance Tasks go to: http://www.collegeboard.com/html/computerscience/index.html?MTG77-ED-1-apcs

Computational Thinking Practices Skills

P1: Connecting computing
Developments in computing have far-reaching effects on society and have led to significant innovations. These developments have implications for individuals, society, commercial markets, and innovation. Students in this course study these effects and connections, and they learn to draw connections between different computing concepts. Students are expected to:

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

P2: Creating computational artifacts
Computing is a creative discipline in which the creation takes many forms, such as remixing digital music, generating animations, developing websites, and writing programs. Students in this course engage in the creative aspects of computing by designing and developing interesting computational artifacts, as well as by applying computing techniques to creatively solve problems. Students are expected to:

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

P3: Abstracting
Computational thinking requires understanding and applying abstraction at multiple levels, such as, privacy in social networking applications, logic gates and bits, the human genome project, etc. 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 are represented for computational use;
- Explain how abstractions are used in computation or modeling;
- Identify abstractions; and
- Describe modeling in a computational context.

P4: Analyzing problems and artifacts
The results and artifacts of computation and the computational techniques and strategies that generate them can be understood both intrinsically for what they are as well as for what they produce. They can also be analyzed and evaluated by applying aesthetic, mathematical, pragmatic, and other criteria. Students in this course design and produce solutions, models, and artifacts, and they evaluate and analyze their own computational work as well as the computational work that others have produced. Students are expected to:

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

P5: Communicating

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

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

P6: Collaborating

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

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

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 these creative aspects of computing. Students in this course will use tools and techniques of computer science to create interesting and relevant artifacts with characteristics that are enhanced by computation.

Essential Questions:

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

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
1.1 Creative development can be an essential process for creating computational artifacts.	**1.1.1** Apply a creative development process when creating computational artifacts. [P2]	**1.1.1A** A creative process in the development of a computational artifact can include but is not limited to employing non-traditional, non-prescribed 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.
1.2 Computing enables people to use creative development processes when using computing tools and techniques to create computational artifacts for creative expression of ideas or to solve a problem.	**1.2.1** Create a computational artifact for creative expression. [P2]	**1.2.1A** A computational artifact is anything created by a human using a computer and can be but is not limited to a program, image, audio, video, presentation, or web page file. **1.2.1B** Creating computational artifacts requires understanding and using software tools and services. **1.2.1C** Computing tools and techniques are used to create computational artifacts and can include but are not limited to programming IDEs, spreadsheets, 3D printers, or text editors. **1.2.1D** A creatively developed computational artifact can be created by using non-traditional, non-prescribed computing techniques. **1.2.1E** Creative expressions in a computational artifact can reflect personal expressions of ideas or interests.

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

1.3 Computing can extend traditional forms of human expression and experience.	**1.3.1** Use computing tools and techniques for creative expression. [P2]	**1.3.1A** Creating digital effects, images, audio, video, and animations has transformed industries. **1.3.1B** Digital audio and music can be created by synthesizing sounds, by sampling existing audio and music, and by recording and manipulating sounds, including layering and looping. **1.3.1C** Digital images can be created by generating pixel patterns, manipulating existing digital images, or combining images. **1.3.1D** Digital effects and animations can be created by using existing software or by modified software that includes functionality to implement the effects and animations. **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 includes examples of abstractions used in modeling the world, managing complexity, and communicating with people as well as with machines. Students in this course will learn to work with multiple levels of abstraction while engaging with computational problems and systems, use models and simulations that simplify complex topics in graphical, textual, and tabular formats, and use snapshots of models and simulation outputs to understand how data is changing, identify patterns, and recognize abstractions.

Essential Questions:

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

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
2.1 A variety of abstractions built	**2.1.1** Describe the variety of abstractions used to represent data. [P3]	**2.1.1A** Digital data is represented by abstractions at different levels. **2.1.1B** At the lowest level all digital data are

upon binary sequences can be used to represent all digital data.		represented by bits. **2.1.1C** At a higher level, bits are grouped to represent abstractions including but not limited to numbers, characters and color. **2.1.1D** Number bases, including binary, decimal, and hexadecimal, are used to represent and investigate digital data. **2.1.1E** At one of the lowest levels of abstraction, digital data is represented in binary (base 2) using only combinations of the digits zero and one. **2.1.1F** Hexadecimal (base 16) is used to represent digital data because the hexadecimal representation uses fewer digits than binary. **2.1.1G** Numbers can be converted from any base to any other base.
	2.1.2 Explain how binary sequences are used to represent digital data. [P5]	**2.1.2A** A finite representation is used to model the infinite mathematical concept of a number. **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. **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. **2.1.2D** The interpretation of a binary sequence depends on how it is used. **2.1.2E** A sequence of bits may represent instructions or data. **2.1.2F** A sequence of bits may represent different types of data in different contexts.
2.2 Multiple levels of abstraction are used to write programs or to create other computational artifacts.	**2.2.1** Develop an abstraction when writing a program or creating other computational artifacts. [P2]	**2.2.1A** The process of developing an abstraction involves removing detail and generalizing functionality. **2.2.1B** An abstraction extracts common features from specific examples in order to generalize concepts. **2.2.1C** An abstraction generalizes functionality with input parameters that allow software reuse.
	2.2.2 Use multiple levels of abstraction to write	**2.2.2A** Software is developed using multiple levels of abstractions such as constants, expressions,

	programs. [P3]	statements, procedures, and libraries. **2.2.2B** Being aware of and using multiple levels of abstraction in developing programs helps to more effectively apply available resources and tools to solve problems.
	2.2.3 Identify multiple levels of abstractions being used when writing programs. [P3]	**2.2.3A** Different programming languages offer different levels of abstraction. **2.2.3B** High-level programming languages provide more abstractions for the programmer and are easier for humans to read and write a program. **2.2.3C** Code in a programming language is often translated into code in another lower level language to be executed on a computer. **2.2.3D** In an abstraction hierarchy, higher levels of abstraction (the most general concepts) would be placed toward the top and the lower level abstractions (the more specific concepts) toward the bottom. **2.2.3E** Binary data is processed by physical layers of computing hardware, including gates, chips, and components. **2.2.3F** A logic gate is a hardware abstraction that is modeled by a Boolean function. **2.2.3G** A chip is an abstraction composed of low-level components and circuits that perform a specific function. **2.2.3H** A hardware component can be low level like a transistor or high level like a video card. **2.2.3I** Hardware is built using multiple levels of abstractions such as transistors, logic gates, chips, memory, motherboards, special purposes cards and storage devices. **2.2.3J** Applications and systems are designed, developed, and analyzed using levels of hardware, software, and conceptual abstractions. **2.2.3K** Lower level abstractions can be combined to make higher level abstractions such as short message services (SMS) or email messages, images, audio files and videos.
2.3 Models and simulations use abstraction to generate new understanding	**2.3.1** Use models and simulations to represent phenomena. [P3]	**2.3.1A** Models and simulations are simplified representations of a more complex objects or phenomena. **2.3.1B** Models may use different abstractions or levels of abstraction depending on the objects or

and knowledge.		phenomena being posed. **2.3.1C** Models often omit unnecessary features of the objects or phenomena that are being modeled. **2.3.1D** Simulations mimic real-world events without the cost or danger of building and testing the phenomenon in the real world.
	2.3.2 Use models and simulations to formulate, refine, and test hypotheses. [P3]	**2.3.2A** Models and simulations facilitate the formulation and refinement of hypotheses related to the object or phenomena under consideration. **2.3.2B** Hypotheses are formulated to explain the object or phenomena being modeled. **2.3.2C** Hypotheses are refined by examining the insights models and simulations provide into the object or phenomena. **2.3.2D** The results of simulations may generate new knowledge and new hypotheses related to the phenomena being modeled. **2.3.2E** Simulations allow hypotheses to be tested without the constraints of the real world. **2.3.2F** Simulations can facilitate extensive and rapid testing of models. **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. **2.3.2H** Rapid and extensive testing allows models to be changed to accurately reflect the object 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 that have led to monumental change across 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 create information. Computation and computer science facilitate and enable a new understanding of data and information that contributes knowledge to the world. Students in this course will work with data using a variety of computational tools [LD8] [f9] [RK10] and techniques to better understand the many ways in which data is transformed into information and knowledge.

Essential Questions:

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

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
3.1 People use computer programs to process information to gain insight and knowledge.	**3.1.1** Use computers to process information, find patterns, and test hypotheses about digitally processed information to gain insight and knowledge. [P4]	**3.1.1A** Computers are used in an iterative and interactive way when processing digital information to gain insight and knowledge. **3.1.1B** Digital information can be filtered and cleaned by using computers to process information. **3.1.1C** Combining data sources, clustering data and data classification are part of the process of using computers to process information. **3.1.1D** Insight and knowledge can be obtained from translating and transforming digitally represented information. **3.1.1E** Patterns can emerge when data is transformed using computational tools.
	3.1.2 Collaborate when processing information to gain insight and knowledge. [P6]	**3.1.2A** Collaboration is an important part of solving data-driven problems. **3.1.2B** Collaboration facilitates solving computational problems through multiple perspectives, experiences, and skill sets. **3.1.2C** Communication between participants working on data-driven problems gives rise to enhanced insights and knowledge. **3.1.2D** Collaboration in developing hypotheses and questions and in testing hypotheses and answering questions about data helps gain insight and knowledge. **3.1.2E** Collaborating face-to-face and using online collaborative tools can facilitate processing information to gain insight and knowledge. **3.1.2F** Investigating large data sets collaboratively can lead to insight and knowledge not obtained working alone.
	3.1.3 Explain the insight and knowledge gained from digitally processed data by	**3.1.3A** Visualization tools and software can communicate information about data. **3.1.3B** Tables, diagrams, and textual displays can be

	using appropriate visualizations, notation, and precise language. [P5]	used in communicating insight and knowledge gained from data. **3.1.3C** Summaries of data analyzed computationally can be effective in communicating insight and knowledge gained from digitally represented information. **3.1.3D** Transforming information can be effective in communicating knowledge gained from data. **3.1.3E** Interactivity with data is an aspect of communicating.
3.2 Computing facilitates exploration and the discovery of connections in information.	**3.2.1** Extract information from data to discover and explain connections, patterns, or trends. [P1]	**3.2.1A** The use of large data sets provides opportunities and challenges for extracting information and knowledge. **3.2.1B** Large data sets provide opportunities for identifying trends, making connections in data, and solving problems. **3.2.1C** Computing tools facilitate the discovery of connections in information within large data sets. **3.2.1D** Search tools are essential for efficiently finding information. **3.2.1E** Information filtering systems are important tools for finding information and recognizing patterns in the information. **3.2.1F** Software tools, including spreadsheets and databases, help to efficiently organize and find trends in information. **3.2.1G** Metadata is data about data. **3.2.1H** Metadata can be descriptive data about an image, a web page, or other complex objects. **3.2.1I** Metadata can increase the effective use of data or data sets by providing additional information about various aspects of that data.
	3.2.2. Use large data sets to explore and discover information and knowledge. [P3]	**3.2.2A** Large data sets include data such as transactions, measurements, text, sound, images, and video. **3.2.2B** The storing, processing, and curating of large data sets is challenging. **3.2.2C** Structuring large data sets for analysis can be challenging. **3.2.2D** Maintaining privacy of large data sets containing personal information can be challenging. **3.2.2E** Scalability of systems is an important consideration when data sets are large. **3.2.2F** The size or scale of a system that stores data

		affects how that data set is used. **3.2.2G** The effective use of large data sets requires computational solutions. **3.2.2H** Analytical techniques to store, manage, transmit, and process data sets change as the size of data sets scale.
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]	**3.3.1A** Digital data representations involve trade-offs related to storage, security, and privacy concerns. **3.3.1B** Security concerns engender trade-offs in storing and transmitting information. **3.3.1C** There are trade-offs in using lossy and lossless compression techniques for storing and transmitting data. **3.3.1D** Lossless data compression reduces the number of bits stored or transmitted, but allows complete reconstruction of the original data. **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. **3.3.1F** Security and privacy concerns arise with data containing personal information. **3.3.1G** Data is stored in many formats depending on its characteristics, such as size and intended use. **3.3.1H** The choice of storage media affects both the methods of and costs of manipulating the data it contains. **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 tasks. 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 is one of the most fundamental aspects of computing. Students in this course will work with algorithms in many ways: they will develop and express original algorithms, they will implement algorithms in some language, and they will analyze algorithms both analytically and empirically.

Essential Questions:

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

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

		for specific domains and are better for expressing algorithms in those domains**.** **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. **4.1.2G** Every algorithm can be constructed using only sequencing, selection, and iteration. **4.1.2H** Nearly all programming languages are equivalent in terms of being able to express any algorithm. **4.1.2I** Clarity and readability are important considerations when expressing an algorithm in a language.
4.2 Algorithms can solve many but not all problems.	**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]	**4.2.1A** Many problems can be solved in a reasonable time. **4.2.1B** Reasonable time means that as the input size grows, the number of steps the algorithm takes is proportional to the square (or cube, fourth power, fifth power, etc.) of the size of the input. **4.2.1C** Some problems cannot be solved in a reasonable time, even for small input sizes. **4.2.1D** Some problems can be solved but cannot be solved in a reasonable time. In these cases, heuristic approaches may be helpful to find solutions in reasonable time.
	4.2.2 Explain the difference between solvable and unsolvable problems in computer science. [P1]	**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. **4.2.2B** Heuristics may be helpful finding an approximate solution more quickly when exact methods are too slow. **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. **4.2.2D** Some problems cannot be solved using any algorithm.
	4.2.3 Explain the existence of undecidable problems in computer science. [P1]	**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. **4.2.3B** A decidable problem is one in which an

		algorithm can be constructed to answer 'yes' or 'no' for all inputs, such as "is the number even?" **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.
	4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity. [P4]	**4.2.4A** Determining an algorithm's efficiency is done by reasoning formally or mathematically about the algorithm. **4.2.4B** Empirical analysis of an algorithm is done by implementing the algorithm and running it on different inputs. **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. **4.2.4D** Different correct algorithms for the same problem can have different efficiencies. **4.2.4E** Sometimes more efficient algorithms are more complex. **4.2.4F** Finding an efficient algorithm for a problem can help solve larger instances of the problem. **4.2.4G** Efficiency includes both execution time and memory usage. **4.2.4H** Linear search can be used when searching for an item in any list but 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 have changed our lives. Programming results in the creation of software, and it facilitates the creation of computational artifacts including music, images, visualizations, and more. In this course, programming will enable exploration and is the object of study. This course will introduce students to the concepts and techniques related to writing programs, developing software, and using software effectively; the focus of the course is not on programming per se, but on all aspects of computation. Students in this course will create programs, translating human intention into computational artifacts.

Essential Questions:

- How are programs developed to help people, organizations or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?

- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- How do people develop and test computer programs?
- Which mathematical and logical concepts are fundamental to computer programming?

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
5.1 Programs can be developed to solve problems (to help people, organizations or society); for creative expression; to satisfy personal curiosity or to create new knowledge.	**5.1.1** Develop a program for creative expression, to satisfy personal curiosity or to create new knowledge. [P2]	**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. **5.1.1B** Programs developed for creative expression or to satisfy personal curiosity may have visual, audible, or tactile inputs and outputs. **5.1.1C** Programs developed for creative expression or to satisfy personal curiosity may be developed with different standards or methods than programs developed for widespread distribution. **5.1.1D** Additional desired outcomes may be realized independently of the original purpose of the program. **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. **5.1.1F** Advances in computing have generated and increased creativity in other fields.
	5.1.2 Develop a correct program to solve problems. [P2]	**5.1.2A** An iterative process of program development helps in developing a correct program to solve problems. **5.1.2B** Developing correct program components and then combining them helps in creating correct programs. **5.1.2C** Incrementally adding tested program segments to correct, working programs helps create large correct programs. **5.1.2D** Program documentation helps programmers develop and maintain correct programs to efficiently solve problems. **5.1.2E** Documentation about program components, such as blocks and procedures help in developing and maintaining programs. **5.1.2F** Documentation helps in developing and maintaining programs when working individually or

		in collaborative programming environments. **5.1.2G** Program development includes identifying programmer and user concerns that affect the solution to problems. **5.1.2H** Consultation and communication with program users is an important aspect of program development to solve problems. **5.1.2I** A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem. **5.1.2J** A programmer designs, implements, tests, debugs, and maintains programs when solving problems.
	5.1.3 Collaborate to develop a program. [P6]	**5.1.3A** Collaboration can decrease the size and complexity of tasks required of individual programmers. **5.1.3B** Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming. **5.1.3C** Collaboration in the iterative development of a program requires different skills than developing a program alone. **5.1.3D** Collaboration can make it easier to find and correct errors when developing programs. **5.1.3E** Collaboration facilitates developing program components independently. **5.1.3F** Effective communication between participants is required for successful collaboration when developing programs.
5.2 People write programs to execute algorithms.	**5.2.1** Explain how programs implement algorithms. [P3]	**5.2.1A** Algorithms are implemented using program instructions that are processed during program execution. **5.2.1B** Program instructions are executed sequentially. **5.2.1C** Program instructions may involve variables that are initialized and updated, read and written. **5.2.1D** An understanding of instruction processing and program execution is useful for programming. **5.2.1E** Program execution automates processes. **5.2.1F** Processes use memory, a central processing unit (CPU), and input and output. **5.2.1G** A process may execute by itself or with other processes. **5.2.1H** A process may execute on one or several

		CPUs. **5.2.1I** Executable programs increase the scale of problems that can be addressed. **5.2.1J** Simple algorithms can solve a large set of problems when automated. **5.2.1K** Improvements in algorithms, hardware, and software increase the kinds of problems and the size of problems solvable by programming.
5.3 Programming is facilitated by appropriate abstractions.	**5.3.1** Use abstraction to manage complexity in programs. [P3]	**5.3.1A** Procedures are reusable programming abstractions. **5.3.1B** A function is a named grouping of programming instructions. **5.3.1C** Procedures reduce the complexity of writing and maintaining programs. **5.3.1D** Procedures have names and may have parameters and return values. **5.3.1E** Parameterization can generalize a specific solution. **5.3.1F** Parameters generalize a solution by allowing a function to be used instead of duplicated code. **5.3.1G** Parameters provide different values as input to procedures when they are called. **5.3.1H** Data abstraction provides a means of separating behavior from implementation. **5.3.1I** Strings and string operations, including concatenation and some form of substring, are common in many programs. **5.3.1J** Integers and floating-point numbers are used in programs without requiring understanding of how they are implemented. **5.3.1K** Lists and list operations such as add, remove, and search are common in many programs. **5.3.1L** Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain. **5.3.1M** Application program interfaces (APIs) and libraries simplify complex programming tasks. **5.3.1N** Documentation for an API/library is an important aspect of programming. **5.3.1O** APIs connect software components, allowing them to communicate.
5.4 Programs are developed,	**5.4.1** Evaluate the correctness of a program.	**5.4.1A** Program style can affect the determination of program correctness.

maintained, and used by people for different purposes.	[P4]	**5.4.1B** Duplicated code can make it harder to reason about a program. **5.4.1C** Meaningful names for variables and procedures help people better understand programs. **5.4.1D** Longer code blocks are harder to reason about than shorter code blocks in a program. **5.4.1E** Locating and correcting errors in a program is called debugging the program. **5.4.1F** Knowledge of what a program is supposed to do is required in order to find most program errors. **5.4.1G** Examples of intended behavior on specific inputs help people understand what a program is supposed to do. **5.4.1H** Visual displays (or different modalities) of program state can help in finding errors. **5.4.1I** Programmers justify and explain a program's correctness. **5.4.1J** Justification can include a written explanation about how a program meets its specifications. **5.4.1K** Correctness of a program depends on correctness of program components, including code blocks and procedures. **5.4.1L** An explanation of a program helps people understand the functionality and purpose of a program. **5.4.1M** The functionality of a program is often described by how a user interacts with the program. **5.4.1N** The functionality of a program is best described at a high level by what the program does, not at a lower level of how the program statements work to accomplish this.
5.5 Programming uses mathematical and logical concepts.	**5.5.1** Employ appropriate mathematical and logical concepts in programming. [P1]	**5.5.1A** Numbers and numerical concepts are fundamental to programming. **5.5.1B** Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations. **5.5.1C** Real numbers are approximated by floating-point representations that do not necessarily have infinite precision. **5.5.1D** Mathematical expressions using arithmetic operators are part of most programming languages.

		5.5.1E Logical concepts and Boolean algebra are fundamental to programming. **5.5.1F** Compound expressions using *and*, *or*, and *not* are part of most programming languages. **5.5.1G** Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs. **5.5.1H** Computational methods may use lists and collections to solve problems. **5.5.1I** Lists and other collections can be treated as abstract data types (ADTs) in developing programs. **5.5.1J** Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection.

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 will gain insight into how the Internet operates, study characteristics of the Internet and systems built upon it, and analyze important concerns such as cybersecurity.

Essential Questions:

- What is the Internet, how is it built, and how does it function?
- What aspects of the Internet's design and development have helped it scale and flourish?
- How is cybersecurity impacting the ever increasing number of Internet users?

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
6.1 The Internet is a network of autonomous systems.	**6.1.1** Explain the abstractions in the Internet and how the Internet functions. [P3]	**6.1.1A** The Internet connects devices and networks all over the world. **6.1.1B** An end-to-end architecture facilitates connecting new devices and networks on the Internet. **6.1.1C** Devices and networks that make up the Internet are connected and communicate using addresses and protocols.

		6.1.1D The Internet and the systems built on it facilitate collaboration. **6.1.1E** Connecting new devices to the Internet is enabled by assignment of an IP address. **6.1.1F** The Internet is built on evolving standards including those for addresses and names. **6.1.1G** The Domain Name System (DNS) translates names to Internet protocol (IP) addresses. **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. **6.1.1I** Standards such as hypertext transfer protocol (HTTP), Internet protocol (IP), and simple mail transfer protocol (SMTP) are developed and overseen by the Internet Engineering Task Force (IETF).
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]	**6.2.1A** The Internet and the systems built on it are hierarchical and redundant. **6.2.1B** The domain name syntax is hierarchical. **6.2.1C** IP addresses are hierarchical. **6.2.1D** Routing on the Internet is fault tolerant and redundant.
	6.2.2 Explain how the characteristics of the Internet influence the systems built on it. [P4]	**6.2.2A** Hierarchy and redundancy help systems scale. **6.2.2B** The duplication of routing (i.e., more than one way to route data) between two points on the Internet increases the reliability of the Internet and helps it scale to more devices and more people. **6.2.2C** Hierarchy in the Domain Name System (DNS) helps that system scale. **6.2.2D** Interfaces and protocols enable widespread use. **6.2.2E** Open standards fuel the growth of the Internet. **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 that contain both the data being transmitted and control information for routing the data. **6.2.2G** Standards for packets and routing include transmission control protocol/Internet protocol (TCP/IP). **6.2.2H** Standards for sharing information and

		communicating between browsers and servers on the web include hypertext transfer protocol (HTTP), and secure sockets layer/transport layer security (SSL/TLS). **6.2.2I** The size and speed of systems affect their use. **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. **6.2.2K** The latency of a system is the time elapsed between the transmission and the receipt of a request.
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]	**6.3.1A** The trust model of the Internet involves trade-offs. **6.3.1B** The Domain Name System (DNS) was not designed to be completely secure. **6.3.1C** Implementing cybersecurity has software, hardware, and human components. **6.3.1D** Cyber warfare and cybercrime have widespread and potentially devastating effects. **6.3.1E** Distributed denial-of-service attacks (DDoS) compromise a target by flooding it with requests from multiple systems. **6.3.1F** Phishing, viruses, and other attacks have human and software components. **6.3.1G** Antivirus software and firewalls can help prevent unauthorized access to private data. **6.3.1H** Cryptography is essential to many models of cybersecurity. **6.3.1I** Cryptography has a mathematical foundation. **6.3.1J** Open standards help ensure cryptography is secure. **6.3.1K** Symmetric encryption is a method of encryption involving one key for encryption and decryption. **6.3.1L** Public key encryption, which is not symmetric, is an encryption method that is widely used because of the enhanced security associated with its use. **6.3.1M** Certificate authorities (CAs) issue digital certificates that validate the ownership of encrypted keys used in secured communication and are based on a trust model.

Big Idea 7: Global Impact.

Computing has global impacts. Computation has changed the way people think, work, live, and play. Our methods for communicating, collaborating, problem solving, and doing business have changed and are changing due to innovations enabled by computing. Many innovations in other fields are fostered by advances in computing. Computational approaches lead to new understandings, new discoveries, and new disciplines. Students in this course will become familiar with many ways in which computing enables innovation, and they will 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	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
7.1 Computing enhances communication, interaction, and cognition.	**7.1.1** Explain how computing innovations affect communication, interaction, and cognition. [P4]	**7.1.1A** Email, short message service (SMS), and chat have fostered new ways to communicate and collaborate. **7.1.1B** Video conferencing and video chat have fostered new ways to communicate and collaborate. **7.1.1C** Social media continues to evolve and foster new ways to communicate. **7.1.1D** Cloud computing fosters new ways to communicate and collaborate. **7.1.1E** Widespread access to information facilitates the identification of problems, development of solutions, and dissemination of results. **7.1.1F** Public data provides widespread access and enables solutions to identified problems. **7.1.1G** Search trends are predictors. **7.1.1H** Social media, including blogs and twitter, have enabled dissemination. **7.1.1I** Global Positioning System (GPS) and related technologies have changed how humans travel, navigate, and find information related to geolocation. **7.1.1J** Sensor networks facilitate new ways of interacting with the environment and with physical systems.

		7.1.1K Smart grids, smart buildings, and smart transportation are changing and facilitating human capabilities. **7.1.1L** Computing contributes to many assistive technologies that enhance human capabilities. **7.1.1M** The Internet and the Web have enhanced methods of and opportunities for communication and collaboration. **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. **7.1.1O** The Internet and the Web have impacted productivity, positively and negatively, in many areas.
	7.1.2 Explain how people participate in a problem solving process that scales. [P4]	**7.1.2A** Distributed solutions must scale to solve some problems. **7.1.2B** Science has been impacted by using scale and "citizen science" to solve scientific problems using home computers in scientific research. **7.1.2C** Human computation harnesses contributions from many humans to solve problems related to digital data and the Web. **7.1.2D** Human capabilities are enhanced by digitally enabled collaboration. **7.1.2E** Some online services use the contributions of many people to benefit both individuals and society. **7.1.2F** Crowdsourcing offers new models for collaboration such as connecting people with jobs and businesses with funding. **7.1.2G** The move from desktop computers to a proliferation of always-on mobile computers is leading to new applications.
7.2 Computing enables innovation in nearly every field.	**7.2.1** Explain how computing has impacted innovations in other fields. [P1]	**7.2.1A** Machine learning and data mining have enabled innovation in medicine, business, and science. **7.2.1B** Scientific computing has enabled innovation in science and business. **7.2.1C** Computing enables innovation by providing access to and sharing of information. **7.2.1D** Open access and Creative Commons have enabled broad access to digital information. **7.2.1E** Open and curated scientific databases have

		benefited scientific researchers. **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. **7.2.1G** Advances in computing as an enabling technology have generated and increased the creativity in other fields.
7.3 Computing has global effects – both beneficial and harmful – on people and society.	**7.3.1** Analyze the beneficial and harmful effects of computing. [P4]	**7.3.1A** Innovations enabled by computing raise legal and ethical concerns. **7.3.1B** Commercial access to music and movie downloads and streaming raises legal and ethical concerns. **7.3.1C** Access to digital content via peer-to-peer networks raises legal and ethical concerns. **7.3.1D** Both authenticated and anonymous access to digital information raises legal and ethical concerns. **7.3.1E** Commercial and governmental censorship of digital information raise legal and ethical concerns. **7.3.1F** Open source and licensing of software and content raise legal and ethical concerns. **7.3.1G** Privacy and security concerns arise in the development and use of computational systems and artifacts. **7.3.1H** Aggregation of information including geolocation, cookies, and browsing history raises privacy and security concerns. **7.3.1I** Anonymity in online interactions can be enabled through the use of online anonymity software and proxy servers. **7.3.1J** Technology enables collection, use, and exploitation of information about, by, and for individuals, groups, and institutions. **7.3.1K** People can have instant access to vast amounts of information online; accessing this information can enable collection of both individual and aggregate data that can be used and collected. **7.3.1L** Commercial and governmental curation of information may be exploited if privacy and other protections are ignored. **7.3.1M** Targeted advertising is used to help individuals, but it can be misused at both individual and aggregate levels. **7.3.1N** Widespread access to digitized information

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

Contributors:

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

*Members of the AP Computer Science Principles Development Committee