The College Board

AP® Computer Science Principles

Draft Curriculum Framework

August 2013

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 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 "big-data" — they analyze it, visualize it, draw conclusions from trends in it — but the course itself does not specify particular tools 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** capture important aspects of the work that computer scientists engage in at the level of competence expected of AP Computer Science Principles students. A practice is a way to coordinate knowledge and skills in order to accomplish a goal or task. The computational thinking practices 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 practice are equally important in Computer Science Principles, each learning objective includes a correlated computational thinking practice.

- 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 **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** or content detail that must be taught in order for students to achieve and demonstrate an understanding of a particular required concept in the course. 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 corresponds to Objective 1.1.1 Use computing tools and techniques to create artifacts [P2]:

  1.1.1.a Computing enables people to create digitally — creating knowledge, tools, expressions of ideas, and solutions to problems.
    - Creating digitally requires understanding and using software tools.
    - Creating digitally can be done by combining and modifying existing artifacts or by creating new artifacts.

  1.1.1.b Computing enables people to translate intention into digital artifacts.
    - A computational artifact is created by human conception using software tools.
    - Examples of computational artifacts include digital music, videos, images, documents, and combinations of these such as infographics, presentations, and Web pages.

Relationship between the Curriculum Framework and the Assessments

The learning objectives will be targets of two different types of assessment: an end-of-course exam and a through-course assessment component. The through-course component is comprised of three performance tasks — separately, these tasks require students to engage with data, programming, and the internet.

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

Like an exam, the performance tasks represent an opportunity designed to gather *evidence* of student learning with regard to the learning objectives. The tasks measure what is taught in the course, including computational thinking skills.

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.

- To download the current version, go to http://www.collegeboard.com/html/computerscience/index.html?MTG77-ED-1-apcs and click on "Performance Tasks."
- Each contains specific requirements and documentation.
- Each contains a list of the learning objectives covered by the task.
- Two of the three tasks — Data and Programming — require collaboration.

Here are some examples of the connection between the curriculum framework and the performance tasks:

- **DATA PERFORMANCE TASK**

Big Idea 1: 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 knowledge. This course will emphasize these creative aspects of computing. Students in this course will create interesting and relevant artifacts with the tools and techniques of computer science.		
Enduring Understandings	**Learning Objectives (What students must be able to do)**	**Essential Knowledge (What students need to know)**
1.1 Computing fosters the creation of artifacts.	**1.1.2** Collaborate in the creation of computational artifacts. [P6]	**1.1.2.a** Collaboration is an essential part of creating computational artifacts. • Collaboration facilitates multiple perspectives in developing computational artifacts. • A computational artifact can reflect collaborative intent.

- **PROGRAMMING PERFORMANCE TASK**

Big Idea 1: 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 knowledge. This course will emphasize these creative aspects of computing. Students in this course will create interesting and relevant artifacts with the tools and techniques of computer science.		
Enduring Understandings	**Learning Objectives (What students must be able to do)**	**Essential Knowledge (What students need to know)**
1.3 Programming is a creative process.	**1.3.1** Use programming as a creative tool. [P2]	**1.3.1.a** Programs can be developed for creative expression or to satisfy personal curiosity. • A program developed for creative expression or to satisfy personal curiosity may have visual, audible, or tactile results; or the program may affect a computer or system without such results. • Programs developed for creative expression or to satisfy personal curiosity may be developed with different standards or methods than programs developed for widespread distribution. • A program or the results of running a program may be shared with others. **1.3.1.b** Programs can be developed to solve problems, create new knowledge, or help people, organizations, or society. • Programs may be developed specifically with the goal of solving a problem, creating new knowledge, or helping people, organizations, or society; however, the goals may be realized independently of the original purpose of the program. • Computer programs and the results of running the programs have widespread impact on individuals, organizations, and society.

- **INTERNET PERFORMANCE TASK**

Big Idea 6: 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.		
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]	**6.1.1.a** The Internet connects devices and networks all over the world. • An end-to-end architecture facilitates connecting new devices and networks on the Internet. • Internet protocol (IP) addresses are allocated to each autonomous system (AS) on the Internet. **6.1.1.b** The Internet and the systems built on it facilitate collaboration. • Connecting new devices to the Internet is enabled by assignment of an IP number. • Cloud computing facilitates collaboration. **6.1.1.c** The Internet is built on evolving standards including those for addresses and names. • The Domain Name System (DNS) translates names to Internet protocol (IP) addresses. • IP addresses use Internet protocol version 4 (IPv4) and Internet protocol version 6 (IPv6) to enable routing. • 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).

Computational Thinking Practices

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: Developing computational artifacts

Computing is a creative discipline in which the creation takes many forms, ranging from remixing digital music and generating animations to developing websites, writing programs, and more. 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 ranging from privacy in social networking applications, to logic gates and bits, to the human genome project, and more. 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. 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; and
- Collaborate at a large scale.

Big Idea 1: 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 knowledge. This course will emphasize these creative aspects of computing. Students in this course will create interesting and relevant artifacts with the tools and techniques of computer science.

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
1.1 Computing fosters the creation of artifacts.	**1.1.1** Use computing tools and techniques to create artifacts. [P2]	**1.1.1.a** Computing enables people to create digitally — creating knowledge, tools, expressions of ideas, and solutions to problems. • Creating digitally requires understanding and using software tools. • Creating digitally can be done by combining and modifying existing artifacts or by creating new artifacts. **1.1.1.b** Computing enables people to translate intention into computational artifacts. • A computational artifact is created by human conception using software tools. • Examples of computational artifacts include digital music, videos, images, documents, and combinations of these such as infographics, presentations, and Web pages.
	1.1.2 Collaborate in the creation of computational artifacts. [P6]	**1.1.2.a** Collaboration is an essential part of creating computational artifacts. • Collaboration facilitates multiple perspectives in developing computational artifacts. • A computational artifact can reflect collaborative intent.
	1.1.3 Analyze computational artifacts. [P4]	**1.1.3.a** A computational artifact can be analyzed for correctness, functionality, and suitability. • A computational artifact may have weaknesses, mistakes, or errors depending on the type of artifact. For example, music created by a program may not have an error but may simply be hard to listen to. • The functionality of a computational artifact may be related to how it is used or how it is perceived. • The suitability (or appropriateness) of a computational artifact may be related to how it is used or how it is perceived.
1.2 Computing fosters creative expression.	**1.2.1** Use computing tools and techniques for creative expression. [P2]	**1.2.1.a** Computing extends traditional forms of human expression and experience. • Computer music can be created by synthesizing

		sounds, by sampling existing music, or by recording and manipulating sounds. • Creating digital effects, images, and animations has impacted and transformed the movie industry. • Computing enables creative exploration of real and synthetic phenomena.
1.3 Programming is a creative process.	**1.3.1** Use programming as a creative tool. [P2]	**1.3.1.a** Programs can be developed for creative expression or to satisfy personal curiosity. • A program developed for creative expression or to satisfy personal curiosity may have visual, audible, or tactile results; or the program may affect a computer or system without such results. • Programs developed for creative expression or to satisfy personal curiosity may be developed with different standards or methods than programs developed for widespread distribution. • A program or the results of running a program may be shared with others. **1.3.1.b** Programs can be developed to solve problems, create new knowledge, or help people, organizations, or society. • Programs may be developed specifically with the goal of solving a problem, creating new knowledge, or helping people, organizations, or society; however, the goals may be realized independently of the original purpose of the program. • Computer programs and the results of running the programs have widespread impact on individuals, organizations, and society.

Big Idea 2: 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 will include 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.

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
2.1 A combination of abstractions built upon binary sequences can be used to represent all digital data.	**2.1.1** Describe the combination of abstractions used to represent data. [P3]	**2.1.1.a** A combination of abstractions is used to represent digital data. • At the lowest level all digital data are represented by bits. • Bits are grouped to represent higher-level abstractions including numbers and characters. • Higher-level abstractions such as Internet protocol (IP) packets, images, and audio files are comprised of groups of bits that represent different parts of the abstractions. **2.1.1.b** Number bases, including binary and decimal, are used for reasoning about digital data. • Bits represent binary data using base two digits: zero and one. • Hexadecimal, or base-16, is often used in reasoning about data such as colors in images. • Different bases help in reasoning about digital data; digital data is stored in bits.
	2.1.2 Explain how binary sequences are used to represent digital data. [P5]	**2.1.2.a** A finite representation is used to model the infinite mathematical concept of a number. • In many programming languages the fixed number of bits used to represent integers limits the range of integer values, and mathematical operations can result in overflow or other errors. • 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 can result in round-off and other errors. **2.1.2.b** The interpretation of a binary sequence depends on how it is used (e.g., as instruction, number, text, sound, or image). • The sequence of bits that represents an instruction may also represent data processed by that instruction. • The sequence of bits that represents a character/letter may also represent a number.

		• The sequence of bits that represents a color in an image may also represent a sound in an audio file.
2.2 Multiple levels of abstraction are used in computation.	**2.2.1** Develop an abstraction. [P2]	**2.2.1.a** Software and hardware are built using multiple levels of abstraction. • Software is built using low- and high-level abstractions such as expressions, statements, data types, functions, and libraries. • Hardware is built using low- and high-level abstractions such as chips, memory, and storage. • Software is built with abstractions that represent hardware, such as device drivers and game controllers.
	2.2.2 Use multiple levels of abstraction in computation. [P3]	**2.2.2.a** Binary data is processed by physical layers of computing hardware, including gates, chips, and components. • A logic gate is a hardware abstraction that models a Boolean function. • A chip is an abstraction composed of low-level components and circuits that performs a specific function such as memory, CPU, encryption, and more. • A hardware component can be low level like a transistor or high level like a video card. **2.2.2.b** Programming languages, from low to high level, are used in developing software. • Low-level programming languages, such as assembly, are closer to the machine level and provide fewer abstractions for the programmer. • High-level programming languages provide more abstractions for the programmer and are easier for humans to use for reading and writing code. • Code in a high-level programming language is typically automatically translated into code in a lower-level language to be executed on a computer; this is done by a compiler or an interpreter. **2.2.2.c** Applications and systems are designed, developed, and analyzed using levels of hardware, software, and conceptual abstractions. • Mobile applications and systems are designed, developed, and analyzed using hardware, software, and conceptual abstractions. • Web services are both an application and a system that are designed, developed, and analyzed using software, hardware, and conceptual abstractions.
2.3 Models and	**2.3.1** Use models and simulations	**2.3.1.a** People use models and simulations to generate

simulations use abstraction to raise and answer questions.	to raise and answer questions. [P3]	new understanding and knowledge. • Models and simulations enable finding information about a phenomenon without building or testing the phenomenon in the real world. • A computer simulation may run in seconds, hours, or weeks depending on the model being simulated. **2.3.1.b** Models use different levels of abstraction to represent phenomena. • Some features of phenomena are typically ignored when creating models. • Two models of the same phenomena may use different abstractions and make different assumptions about the modeled phenomena. **2.3.1.c** Models and simulations are used to formulate, refine, and test hypotheses. • Hypotheses are refined by noting how the model and simulations explain observations of the phenomena being modeled. • Running simulations may generate new knowledge and new hypotheses related to the phenomena being modeled. • Simulations allow hypotheses to be tested without the constraints of the real world. **2.3.1.d** Simulations can facilitate extensive and rapid testing of models. • Some simulations can be done more quickly with improvements in models, software, and hardware. • Rapid and extensive testing allows models to be changed as part of answering questions accurately.

Big Idea 3: 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 tools and techniques to better understand the many ways in which data is transformed into information and 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 to gain insight and knowledge. [P1]	**3.1.1.a** Computers can be used to find patterns in, answer questions about, and test hypotheses about digitally represented information. • Digital information is filtered and cleaned as part of using computers to process information. • Clustering and classification are part of the process of using computers to process information. • An iterative and interactive engagement with digital information is part of using computers to process information. **3.1.1.b** Insight and knowledge can be obtained from translating and transforming digitally represented information. • Patterns can emerge when data is transformed using computational tools.
	3.1.2 Collaborate when processing information to gain insight and knowledge. [P6]	**3.1.2.a** Collaboration is an essential part of solving data-driven problems. • Collaboration facilitates solving computational problems through multiple perspectives, experiences, and skill sets. • Communication between participants working on data-driven problems gives rise to better insights and deeper knowledge. **3.1.2.b** Collaboration in developing hypotheses and questions and in testing hypotheses and answering questions about data helps gain insight and knowledge. • Collaboration can take place with participants working closely together, or it can take place with participants using online communities. • Managing Big Data collaboratively provides insight and knowledge that cannot be obtained working alone.
	3.1.3 Communicate insight and knowledge gained from using	**3.1.3.a** Visualizations, notation, and precise language are essential in communicating insight and knowledge gained

	computer programs to process information. [P5]	from data. • Visualization tools and software can connect data with communication of information. • Tables, diagrams, and other textual displays should be used in communicating insight and knowledge gained from data. **3.1.3.b** Summaries of insight and knowledge are effective in communicating insights gained from digitally represented information. • Transforming information is effective in communicating (e.g., scaling in charts and tables). • Interactivity is an aspect of communicating (e.g., by filtering displayed data or hiding components of what is shown).
3.2 Computing facilitates exploration and the discovery of connections in information.	**3.2.1** Use computing to facilitate exploration and the discovery of connections in information. [P1]	**3.2.1.a** Computing tools facilitate the discovery of connections in information and extracting information and knowledge from data. • Search tools are essential tools in finding information. • Collaborative filtering is an essential tool in finding information. • Software tools, including spreadsheets and databases, help in finding information. **3.2.1.b** Metadata can increase the effective use of data or a data set by providing additional information about various aspects of that data. • Image metadata can include time and geolocation, as well as information about the image such as size and color table. • Standards for metadata facilitate effective use of data depending on the type of data.
	3.2.2. Use large data sets to explore and discover information and knowledge. [P3]	**3.2.2.a** *Big Data* (the use of large data sets) provides opportunities and challenges for extracting information and knowledge. • Big Data offers opportunities for identifying trends, making connections in data, and solving problems. • Big Data includes data such as transactions, measurements, text, sound, images, and video. • Storing, processing, and curating large data sets is a challenge. • Structuring large data sets for analysis is a challenge. • Maintaining privacy of large data sets containing personal information is a challenge. **3.2.2.b** Scalability, of systems and analytical approaches,

		is an important consideration when data sets are large. • Storage systems from solid-state drives to cloud storage affect how Big Data is used. • Use of Big Data requires innovation in software solutions. • Analytical techniques change as the size of data sets scale.
3.3 Computational manipulation of information requires consideration of representation, storage, security, and transmission.	**3.3.1** Analyze the considerations involved in the computational manipulation of information. [P4]	**3.3.1.a** There are trade-offs in representing information as digital data. • There are trade-offs in using lossy and lossless compression techniques for visual and audio information. • Security concerns engender trade-offs in storing and transmitting information. • Privacy concerns arise with some data (e.g., with health and financial information). **3.3.1.b** Data is stored in many formats depending on its characteristics — such as size and intended use — so that it can be manipulated computationally. • Data may be digitally archived for future use. • Storage media affects methods and costs of manipulating data. • Reading data and updating data have different requirements for storage.

Big Idea 4: 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. 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.

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
4.1 An algorithm is a precise sequence of instructions for a process that can be executed by a computer.	**4.1.1** Develop an algorithm designed to be implemented to run on a computer. [P2]	**4.1.1.a** Sequencing, selection, iteration, and recursion are building blocks of algorithms. • Sequencing is the application of each step of an algorithm in the order in which the statements are given. • Selection uses a Boolean condition to determine which of two parts of an algorithm is used. • Iteration is the repetition of a part of an algorithm until a condition is met or for a specified number of times. • Recursion is the repeated application of an algorithm to smaller instances of a problem. (Developing algorithms using recursion is not required.) **4.1.1.b** Algorithms can be combined to make new algorithms. • Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct. • Knowledge of standard algorithms (e.g., search in a list) can help in constructing new algorithms. **4.1.1.c** Different algorithms can be developed to solve the same problem. • Algorithms that solve the same problem can have different efficiencies. • Developing a new algorithm to solve a problem can yield insight into the problem.

4.2 Algorithms are expressed using languages.	**4.2.1** Express an algorithm in a language. [P5]	**4.2.1.a** Languages for algorithms include natural language, pseudocode, and visual and textual programming languages. • Natural language and pseudocode describe algorithms so that humans can understand them. • Algorithms described in programming languages can be executed on a computer. **4.2.1.b** Different languages are better suited for expressing different algorithms. • Natural language may be better than programming languages for expressing algorithms at a high level. • Some programming languages are designed for specific domains and are better for expressing algorithms in those domains (e.g., Structured Query Language (SQL) for making database queries). **4.2.1.c** The language used to express an algorithm can affect characteristics such as clarity or readability but not whether an algorithmic solution exists. • Every algorithm can be constructed using only sequencing, selection, and iteration. • Nearly all programming languages are equivalent in terms of being able to express any algorithm. • Clarity and readability are dependent on the experience of the person reading an algorithm expressed in a language.
4.3 Algorithms can solve many but not all problems.	**4.3.1** Appropriately connect problems and potential algorithmic solutions. [P1]	**4.3.1.a** Many problems can be solved in a reasonable time. • Reasonable time typically means polynomial in the size of the input. • Searching for an item in a list can be done with binary search when the list is sorted; otherwise it can be done with linear search. • Some problems cannot be solved in a reasonable time, even for small input sizes. **4.3.1.b** Some problems can be solved but heuristic approaches are necessary to solve them in a reasonable time. • Some optimization problems (e.g., the traveling salesperson problem or TSP) such as "find the best" or "find the smallest" cannot be solved in a reasonable time, but approximations to the optimal solution can be found in reasonable time.

		• One of the most important unsolved problems in computer science is whether P = NP (P is polynomial-time algorithms, NP is nondeterministic polynomial-time algorithms). • Some algorithms, such as finding the best move in a game, are solved in reasonable time using heuristics. **4.3.1.c** Some problems cannot be solved using any algorithm. • The halting problem is an example of a problem that can be proven to have no algorithmic solution; such problems are called undecidable problems. • An undecidable problem may have instances that have an algorithmic solution, but there is no algorithmic solution that solves all instances of the problem.
4.4 Algorithms are evaluated analytically and empirically.	**4.4.1** Evaluate algorithms analytically and empirically. [P4]	**4.4.1.a** Algorithms can be evaluated using many criteria, including efficiency, correctness, and clarity. • Determining an algorithm's efficiency is usually done by reasoning formally or mathematically about the algorithm. • Empirical analysis of an algorithm is done by implementing the algorithm and running it on different inputs. • The correctness of an algorithm is determined by reasoning formally or mathematically about the algorithm, not by testing an implementation of the algorithm. **4.4.1.b** Different correct algorithms for the same problem can have different efficiencies. • Sometimes more efficient algorithms are more complex. • Finding a new, more efficient algorithm for a problem helps solve larger instances of the problem.

Big Idea 5: 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 more general computational artifacts including music, images, visualizations, and more. In this course, programming will enable exploration and the object of study. This course will introduce students to the concepts and techniques used in writing programs and to the ways in which programs are developed and used by people; 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.

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
5.1 Programs are written to execute algorithms.	**5.1.1** Explain how programs implement algorithms. [P3]	**5.1.1.a** Programming instructions are processed during program execution. • An intuitive understanding of instruction processing is useful for programming. • Programming can be understood by sequential execution using a fetch-execute cycle in a von Neumann architecture. • Variables are read and written, as well as initialized and updated. • Other execution models and more abstract execution models offer utility in implementing algorithms (e.g., MapReduce). **5.1.1.b** Program execution automates processes. • Processes use memory, a central processing unit (CPU), and input and output. • A process may execute by itself or with other processes. • A process may execute on one or several CPUs. **5.1.1.c** Executable programs increase the scale of problems that can be addressed. • Simple algorithms can solve a large set of problems when automated. • Improvements in algorithms, hardware, and software increase the kinds of problems and the size of problems solvable by programming.
5.2 Programming is facilitated by appropriate abstractions.	**5.2.1** Use abstraction to manage complexity in programs. [P3]	**5.2.1.a** Functions are reusable programming abstractions. • A function is a named grouping of programming instructions. • Functions reduce the complexity of writing and maintaining programs. • Functions have names and parameters, and may return values. **5.2.1.b** Parameterization can generalize a specific solution.

		• Parameters generalize a solution by allowing a function to be used instead of duplicated code. • Parameters provide different values as input to functions when they are called. **5.2.1.c** Data abstraction provides a means of separating behavior from implementation. • Strings and string operations, including concatenation and some form of substring, are common in many programs. • Integers and floating-point numbers are used in programs without requiring understanding of how they are implemented. • Container abstract data types (ADTs), such as a set or list, enable programming. **5.2.1.d** Application program interfaces (APIs) and libraries simplify complex programming tasks. • Documentation for an API/library is an essential aspect of programming. • APIs connect software components, allowing them to communicate.
5.3 Programs are developed and used by people.	**5.3.1** Evaluate a program for correctness. [P4]	**5.3.1.a** Program style affects the determination of program correctness. • Duplicated code can make it harder to reason about a program. • Meaningful names for variables and functions help in reasoning about programs. • Longer code blocks are harder to reason about than shorter code blocks in a program. **5.3.1.b** Locating and correcting errors in a program is called debugging the program. • Knowledge of what a program is supposed to do, including with specific inputs, helps in finding errors. • Visual displays (or different modalities) of program state can help in finding errors. **5.3.1.c** Programmers justify and explain a program's correctness. • Justification can include a written explanation about how a program meets its specification. • Correctness of a program depends on correctness of program components, including code blocks and functions. **5.3.1.d** Programmers explain how a program functions. • The functionality of a program is often described by how a user interacts with the program.

		• 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.
	5.3.2 Develop a correct program. [P2]	**5.3.2.a** An iterative process of program development helps in developing a correct program. • Incremental additions to a correct, working program help create large correct programs. • Developing (correct) program components independently and then combining them helps in creating correct programs. **5.3.2.b** Program documentation helps programmers develop and maintain correct programs. • Documentation about program components, such as blocks and functions, helps in developing programs. • Documentation helps when more than one programmer is developing a program. • Conventions are often followed in writing documentation, depending both on who a program is written for and what language the program is written in. **5.3.2.c** Program development includes identifying programmer and user concerns. • Consultation and communication with program users is an essential aspect of program development. • A programmer's knowledge and skill affects how a program is developed.
	5.3.3 Collaborate to solve a problem using programming. [P6]	**5.3.3.a** Collaboration is an essential part of writing programs to solve problems. • Collaboration can increase the size and complexity of a program, making it possible to address more problems. • Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming. **5.3.3.b** Collaboration in the iterative development of a program requires different skills than developing a program alone. • Collaboration can make it easier to find and correct errors when developing programs. • Collaboration facilitates developing program components independently; thus more quickly developing a program. • Communication between participants is required

		for successful collaboration.
5.4 Programming uses mathematical and logical concepts.	**5.4.1** Employ appropriate mathematical and logical concepts in programming. [P1]	**5.4.1.a** Numbers and numerical concepts are fundamental to programming. • Integers may be constrained in the maximum and minimal values that can be represented in a program because of storage limitations. • Real numbers are approximated by floating-point representations that do not have infinite precision. • Mathematical expressions using arithmetic operators are part of most programming languages. **5.4.1.b** Logical concepts and Boolean algebra are fundamental to programming. • Compound expressions using *and*, *or*, and *not* are part of most programming languages. • Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs. **5.4.1.c** Sets and collections are tools for solving computational problems. • Sets, lists, and other collections can be treated as abstract data types (ADTs) in developing programs. • 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 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.

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]	**6.1.1.a** The Internet connects devices and networks all over the world. • An end-to-end architecture facilitates connecting new devices and networks on the Internet. • Internet protocol (IP) addresses are allocated to each autonomous system (AS) on the Internet. **6.1.1.b** The Internet and the systems built on it facilitate collaboration. • Connecting new devices to the Internet is enabled by assignment of an IP number. • Cloud computing facilitates collaboration. **6.1.1.c** The Internet is built on evolving standards including those for addresses and names. • The Domain Name System (DNS) translates names to Internet protocol (IP) addresses. • IP addresses use Internet protocol version 4 (IPv4) and Internet protocol version 6 (IPv6) to enable routing. • 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 and the systems built on it influence their use.	**6.2.1** Explain characteristics of the Internet and the systems built on it. [P5]	**6.2.1.a** The Internet and the systems built on it are hierarchical and redundant. • The domain name syntax (e.g., dept.college.edu) is hierarchical. • IP addresses in both IPv4 and IPv6 are hierarchical. • Routing on the Internet is fault tolerant and redundant.
	6.2.2 Analyze how characteristics of the Internet and the systems built on it influence their use. [P4]	**6.2.2.a** Hierarchy and redundancy help systems scale. • 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. • Hierarchy in the Domain Name System (DNS) helps that system scale. **6.2.2.b** Interfaces and protocols enable widespread use. • Open standards developed by the Internet Engineering Task Force (IETF) and other groups fuel the growth of the Internet. • Standards for packets and routing include transmission control protocol/Internet protocol (TCP/IP). • Standards for sharing information include hypertext transfer protocol (HTTP), simple mail transfer protocol (SMTP), secure sockets layer/transport layer security (SSL/TLS). **6.2.2.c** The size and speed of systems affect their use. • 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. • The latency of a system is a measure of time, such as the time to send a packet from one point in the system to another.
6.3 Cybersecurity is an important concern for the Internet and the systems built on it.	**6.3.1** Connect the concern of cybersecurity with the Internet and the systems built on it. [P1]	**6.3.1.a** The trust model of the Internet involves trade-offs. • Certificate authorities (CAs) are based on a trust model. • The Domain Name System (DNS) was not designed to be completely secure and is vulnerable to attack. • Domain Name System Security Extensions (DNSSEC) are protocols that add digital signatures to DNS to provide a layer of security. **6.3.1.b** Implementing cybersecurity has software, hardware, and human components. • Cyber warfare threats include espionage and sabotage. • Distributed denial-of-service attacks (DDoS) compromise a target by flooding it with requests from multiple systems. • Phishing, viruses, and other attacks have human and software components. **6.3.1.c** Cryptography is essential to many models of cybersecurity. • One-way functions (e.g., factoring the product of two large primes) are used in cryptography. • Open standards help ensure cryptography is secure.

		• Encryption methods include symmetric, asymmetric, and public key.

Big Idea 7: 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.

Enduring Understandings	Learning Objectives (What students must be able to do)	Essential Knowledge (What students need to know)
7.1 Computing affects communication, interaction, and cognition.	**7.1.1** Analyze how computing affects communication, interaction, and cognition. [P4]	**7.1.1.a** Computing enhances communication, fostering new ways to communicate and collaborate. • Email, short message service (SMS), and chat have fostered new ways to communicate and collaborate. • Video conferencing and video chat have fostered new ways to communicate and collaborate. • Twitter, blogs, and social media have fostered new ways to communicate. • Cloud computing fosters new ways to communicate and collaborate. **7.1.1.b** Widespread access to information facilitates the identification of problems, development of solutions, and dissemination of results. • Widespread access to sites such as Wikipedia has enabled dissemination. • Search trends are predictors. • Social media, including blogs and twitter, have enabled dissemination. • Public data (e.g., the Open Government Initiative) provides widespread access and enables solutions. **7.1.1.c** Computing enhances human capabilities. • Global Positioning System (GPS) and related technologies have changed how humans travel, navigate, and find information related to geolocation. • Sensor networks facilitate new ways of interacting with the environment and with physical systems. • Smart grids, smart buildings, and smart transportation are changing and facilitating human capabilities. • Assistive technologies make computing accessible to a broader human population.

		7.1.1.d The Internet and the Web have a profound impact on society. • The Internet and the Web have enhanced methods of and opportunities for communication and collaboration. • The Internet and the Web have changed many areas, including e-commerce, health care, access to information and entertainment, and online learning. • The Internet and the Web have impacted productivity in many areas.
	7.1.2 Collaborate as part of a process that scales. [P6]	**7.1.2.a** Distributed solutions must scale to solve some problems. • Science has been impacted by using scale and "citizen science" to solve scientific problems using home computers in scientific research (e.g., folding@home, Galaxy Zoo). • Human computation, such as Games With a Purpose, harnesses contributions from many humans to solve problems related to digital data and the Web. **7.1.2.b** Human capabilities are enhanced by computationally enabled collaboration. • reCAPTCHA and Duolingo are examples of services that use the contributions of many people to benefit both individuals and society. • Crowdsourcing offers new business models and opportunities to connect people and jobs (e.g., Mechanical Turk).
7.2 Computing enables innovation in nearly every field.	**7.2.1** Connect computing with innovations in other fields. [P1]	**7.2.1.a** Computational approaches and data analysis enable innovation. • Machine learning and data mining have enabled innovation in medicine, business, and science. • Scientific computing has enabled innovation in science and business. **7.2.1.b** Computing enables innovation by providing access to and sharing of information. • Open access and creative commons have enabled broad access to digital information. • Open and curated scientific databases, such as GenBank and Ensembl, have benefited scientific researchers.
7.3 Computing has both beneficial and harmful effects.	**7.3.1** Analyze the beneficial and harmful effects of computing. [P4]	**7.3.1.a** Innovations enabled by computing raise legal and ethical concerns. • Commercial access to music and movie downloads and streaming raises legal and ethical

		concerns. • Access to digital content via peer-to-peer networks raises legal and ethical concerns. • Both authenticated and anonymous access to digital information raise legal and ethical concerns. • Commercial and governmental censorship of digital information raise legal and ethical concerns. • Open source and licensing of software and content raise legal and ethical concerns. **7.3.1.b** Privacy and security concerns arise in the development and use of computational systems and artifacts. • Aggregation of information including geolocation, cookies, and browsing history causes privacy and security concerns. • Anonymity in online interactions can be enabled by services, such as proxy servers and TOR. **7.3.1.c** Technology enables collection, use, and exploitation of information about, by, and for individuals, groups, and institutions. • 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. • Commercial and governmental curation of information may be exploited if privacy and other protections are ignored. • Targeted advertising is used to help individuals, but it can be misused at both individual and aggregate levels. **7.3.1.d** Widespread access to digitized information raises questions about intellectual property. • Creation of digital audio, video, and textual content by combining existing content has been impacted by copyright concerns. • The Digital Millennium Copyright Act (DMCA) has been a benefit and a challenge in making copyrighted digital material widely available. • Open source and free software have practical, business, and ethical impacts on widespread access to programs, libraries, and code.
7.4 Computing is situated within economic, social, and cultural contexts.	**7.4.1** Connect computing within economic, social, and cultural contexts. [P1]	**7.4.1.a** Computing innovations influence and are influenced by the economic, social, and cultural contexts in which they are designed and used. • The innovation and impact of social media and

		online access is different in different countries and in different socioeconomic groups. • Mobile, wireless, and networked computing have an impact on innovation throughout the world. **7.4.1.b** The global distribution of computing resources raises issues of equity, access, and power. • Groups and individuals are affected by the "digital divide" — differing access to computing and the Internet based on socioeconomic or geographic characteristics. • Networks and infrastructure are supported by both commercial and governmental initiatives.