Georgia CTAE

AP Computer Science Principles (11.01900) (2021)

Adopted 2021

Computational Thinking Practices

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

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

P3. **Computational thinking requires understanding and applying abstraction at multiple levels, such as privacy in social networking applications, logic gates and bits, and the human genome project. Students in this course use abstraction to develop models and simulations of natural and artificial phenomena, use them to make predictions about the world, and analyze their efficacy and validity.** P3

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

P5. **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.** P5

P6. **Innovation can occur when people work together or independently. People working collaboratively can often achieve more than individuals working alone. Learning to collaborate effectively includes drawing on diverse perspectives, skills, and the backgrounds of peers to address complex and open-ended problems. Students in this course collaborate on a number of activities, including the investigation of questions using data sets and the production of computational artifacts.** P6

Big Ideas

BI1. Computing is a creative activity. Creativity and computing are prominent forces in innovation; the innovations enabled by computing have had and will continue to have far-reaching impact. At the same time, computing facilitates exploration and the creation of computational artifacts and new knowledge that help people solve personal, societal, and global problems. This course emphasizes the creative aspects of computing. Students in this course use the tools and techniques of computer science to create interesting and relevant artifacts with characteristics that are enhanced by computation. BI1

1. Creative development can be an essential process for creating computational artifacts. EU 1.1

    1. Apply a creative development process when creating computational artifacts. LO 1.1.1

        1A. A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, non-prescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities. EK 1.1.1A

        1B. Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form. EK 1.1.1B

    2. Computing enables people to use creative development processes to create computational artifacts for creative expression or to solve a problem. EU 1.2

    1. Create a computational artifact for creative expression. LO 1.2.1

        1A. A computational artifact is something created by a human using a computer and can be, but is not limited to, a program, an image, an audio, a video, a presentation, or a Web page file. EK 1.2.1A

        1B. Creating computational artifacts requires understanding of and use of software tools and services. EK 1.2.1B

        1C. Computing tools and techniques are used to create computational artifacts and can include, but are not limited to, programming integrated development environments (IDEs), spreadsheets, three-dimensional (3-D) printers, or text editors. EK 1.2.1C

        1D. A creatively developed computational artifact can be created by using nontraditional, nonprescribed computing techniques. EK 1.2.1D

        1E. Creative expressions in a computational artifact can reflect personal expressions of ideas or interests. EK 1.2.1E

    2. Create a computational artifact using computing tools and techniques to solve a problem. LO 1.2.2

        2A. Computing tools and techniques can enhance the process of finding a solution to a problem. EK 1.2.2A

        2B. A creative development process for creating computational artifacts can be used to solve problems when traditional or prescribed computing techniques are not effective. EK 1.2.2B

    3. Create a new computational artifact by combining or modifying existing artifacts. LO 1.2.3

3A. Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts. EK 1.2.3A

3B. Computation facilitates the creation and modification of computational artifacts with enhanced detail and precision. EK 1.2.3B

3C. Combining or modifying existing artifacts can show personal expression of ideas. EK 1.2.3C

4. Collaborate in the creation of computational artifacts. LO 1.2.4

4A. A collaboratively created computational artifact reflects effort by more than one person. EK 1.2.4A

4B. Effective collaborative teams consider the use of online collaborative tools. EK 1.2.4B

4C. Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation. EK 1.2.4C

4D. Effective collaboration strategies enhance performance. EK 1.2.4D

4E. Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts. EK 1.2.4E

4F. A collaboratively created computational artifact can reflect personal expressions of ideas. EK 1.2.4F

5. Analyze the correctness, usability, functionality, and suitability of computational artifacts. LO 1.2.5

5A. The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact. EK 1.2.5A

5B. A computational artifact may have weaknesses, mistakes, or errors depending on the type of artifact. EK 1.2.5B

5C. The functionality of a computational artifact may be related to how it is used or perceived. EK 1.2.5C

5D. The suitability (or appropriateness) of a computational artifact may be related to how it is used or perceived. EK 1.2.5D

3. Computing can extend traditional forms of human expression and experience. EU 1.3

1. Use computing tools and teachniques for creative expression. LO 1.3.1

1A. Creating digital effects, images, audio, video, and animations has transformed industries. EK 1.3.1A

1B. Digital audio and music can be created by synthesizing sounds, sampling existing audio and music, and recording and manipulating sounds, including layering and looping. EK 1.3.1B

1C. Digital images can be created by generating pixel patterns, manipulating existing digital images, or combining images. EK 1.3.1C

1D. Digital effects and animations can be created by using existing software or modified software that includes functionality to implement the effects and animations. EK 1.3.1D

1E. Computing enables creative exploration of both real and virtual phenomena. EK 1.3.1E

BI2. Abstraction reduces information and detail to facilitate focus on relevant concepts. Everyone uses abstraction on a daily basis to effectively manage complexity. In computer science, abstraction is a central problem-solving technique. It is a process, a strategy, and the result of reducing detail to focus on concepts relevant to understanding and solving problems. This course requires students to use abstractions to model the world and communicate with people as well as with machines. Students in this course learn to work with multiple levels of abstraction while engaging with computational problems and systems; use models and simulations that simplify complex topics in graphical, textual, and tabular formats; and use snapshots of models and simulation outputs to understand how data changes, identify patterns, and recognize abstractions. BI2

1. A variety of abstractions built on binary sequences can be used to represent all digital data. EU 2.1
    1. Describe the variety of abstractions used to represent data. LO 2.1.1
        - 1A. Digital data is represented by abstractions at different levels. EK 2.1.1A
        - 1B. At the lowest level, all digital data are represented by bits. EK 2.1.1B
        - 1C. At a higher level, bits are grouped to represent abstractions, including but not limited to numbers, characters, and color. EK 2.1.1C
        - 1D. Number bases, including binary, decimal, and hexadecimal, are used to represent and investigate digital data. EK 2.1.1D
        - 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. EK 2.1.1E
        - 1F. Hexadecimal (base 16) is used to represent digital data because hexadecimal representation uses fewer digits than binary. EK 2.1.1F
        - 1G. Numbers can be converted from any base to any other base. EK 2.1.1G
    2. Explain how binary sequences are used to represent digital data. LO 2.1.2
        - 2A. A finite representation is used to model the infinite mathematical concept of a number. EK 2.1.2A
        - 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. EK 2.1.2B
        - 2C. In many programming languages, the fixed number of bits used to represent real numbers (as floating- point numbers) limits the range of floating-point values and mathematical operations; this limitation can result in round-off and other errors. EK 2.1.2C
        - 2D. The interpretation of a binary sequence depends on how it is used. EK 2.1.2D
        - 2E. A sequence of bits may represent instructions or data. EK 2.1.2E

2F. A sequence of bits may represent different types of data in different contexts. EK 2.1.2F

2. Multiple levels of abstraction are used to write programs or create other computational artifacts. EU 2.2

   1. Develop an abstraction when writing a program or creating other computational artifiacts. LO 2.2.1

      1A. The process of developing an abstraction involves removing detail and generalizing functionality. EK 2.2.1A

      1B. An abstraction extracts common features from specific examples in order to generalize concepts. EK 2.2.1B

      1C. An abstraction generalizes functionality with input parameters that allow software reuse. EK 2.2.1C

   2. Use multiple levels of abstraction to write programs. LO 2.2.2

      2A. Software is developed using multiple levels of abstractions, such as constants, expressions, statements, procedures, and libraries. EK 2.2.2A

      2B. Being aware of and using multiple levels of abstractions in developing programs help to more effectively apply available resources and tools to solve problems. EK 2.2.2B

   3. Identify multiple levels of abstractions that are used when writing programs. LO 2.2.3

      3A. Different programming languages offer different levels of abstraction. EK 2.2.3A

      3B. High-level programming languages provide more abstractions for the programmer and make it easier for people to read and write a program EK 2.2.3B

      3C. Code in a programming language is often translated into code in another (lower-level) language to be executed on a computer. EK 2.2.3C

      3D. In an abstraction hierarchy, higher levels of abstraction (the most general concepts) would be placed toward the top and lower-level abstractions (the more specific concepts) toward the bottom. EK 2.2.3D

      3E. Binary data is processed by physical layers of computing hardware, including gates, chips, and components. EK 2.2.3E

      3F. A logic gate is a hardware abstraction that is modeled by a Boolean function. EK 2.2.3F

      3G. A chip is an abstraction composed of low-level components and circuits that perform a specific function. EK 2.2.3G

      3H. A hardware component can be low level like a transistor or high level like a video card. EK 2.2.3H

      3I. Hardware is built using multiple levels of abstractions, such as transistors, logic gates, chips, memory, motherboards, special purpose cards, and storage devices. EK 2.2.3I

3J. Applications and systems are designed, developed, and analyzed using levels of hardware, software, and conceptual abstractions. EK 2.2.3J

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. EK 2.2.3K

3. Models and simulations use abstraction to generate new understanding and knowledge. EU 2.3

1. Use models and simulations to represent phenomena. LO 2.3.1

1A. Models and simulations are simplified representations of more complex objects or phenomena. EK 2.3.1A

1B. Models may use different abstractions or levels of abstraction depending on the objects or phenomena being posed. EK 2.3.1B

1C. Models often omit unnecessary features of the objects or phenomena that are being modeled. EK 2.3.1C

1D. Simulations mimic real-world events without the cost or danger of building and testing the phenomena in the real world. EK 2.3.1D

2. Use models and simulations to formulate, refine, and test hypotheses. LO 2.3.2

2A. Models and simulations facilitate the formulation and refinement of hypotheses related to the objects or phenomena under consideration. EK 2.3.2A

2B. Hypotheses are formulated to explain the objects or phenomena being modeled. EK 2.3.2B

2C. Hypotheses are refined by examining the insights that models and simulations provide into the objects or phenomena. EK 2.3.2C

2D. The results of simulations may generate new knowledge and new hypotheses related to the phenomena being modeled. EK 2.3.2D

2E. Simulations allow hypotheses to be tested without the constraints of the real world. EK 2.3.2E

2F. Simulations can facilitate extensive and rapid testing of models. EK 2.3.2F

2G. The time required for simulations is impacted by the level of detail and quality of the models and the software and hardware used for the simulation. EK 2.3.2G

2H. Rapid and extensive testing allows models to be changed to accurately reflect the objects or phenomena being modeled. EK 2.3.2H

BI3. Data and Information BI3

1. People use computer programs to process information to gain insight and knowledge. EU 3.1
    1. Find patterns and test hypotheses about digitally processed information to gain insight and knowledge. LO 3.1.1
        - 1A. Computers are used in an iterative and interactive way when processing digital information to gain insight and knowledge. EK 3.1.1A
        - 1B. Digital information can be filtered and cleaned by using computers to process information. EK 3.1.1B
        - 1C. Combining data sources, clustering data, and data classification are part of the process of using computers to process information. EK 3.1.1C
        - 1D. Insight and knowledge can be obtained from translating and transforming digitally represented information. EK 3.1.1D
        - 1E. Patterns can emerge when data is transformed using computational tools. EK 3.1.1E
    2. Collaborate when Processing information to gain insight and knowledge. LO 3.1.2
        - 2A. Collaboration is an important part of solving data- driven problems. EK 3.1.2A
        - 2B. Collaboration facilitates solving computational problems by applying multiple perspectives, experiences, and skill sets. EK 3.1.2B
        - 2C. Communication between participants working on data-driven problems gives rise to enhanced insights and knowledge. EK 3.1.2C
        - 2D. Collaboration in developing hypotheses and questions, and in testing hypotheses and answering questions, about data helps participants gain insight and knowledge. EK 3.1.2D
        - 2E. Collaborating face-to-face and using online collaborative tools can facilitate processing information to gain insight and knowledge. EK 3.1.2E
        - 2F. Investigating large data sets collaboratively can lead to insight and knowledge not obtained when working alone. EK 3.1.2F
    3. Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notations, and precise language. LO 3.1.3
        - 3A. Visualization tools and software can communicate information about data. EK 3.1.3A
        - 3B. Tables, diagrams, and textual displays can be used in communicating insight and knowledge gained from data. EK 3.1.3B
        - 3C. Summaries of data analyzed computationally can be effective in communicating insight and knowledge gained from digitally represented information. EK 3.1.3C

3D. Transforming information can be effective in communicating knowledge gained from data. EK 3.1.3D

3E. Interactivity with data is an aspect of communicating. EK 3.1.3E

2. Computing facilitates exploration and the discovery of connections in information. EU 3.2

1. Extract information from data to discover and explain connections or trends. LO 3.2.1

1A. Large data sets provide opportunities and challenges for extracting information and knowledge. EK 3.2.1A

1B. Large data sets provide opportunities for identifying trends, making connections in data, and solving problems. EK 3.2.1B

1C. Computing tools facilitate the discovery of connections in information within large data sets. EK 3.2.1C

1D. Search tools are essential for efficiently finding information. EK 3.2.1D

1E. Information filtering systems are important tools for finding information and recognizing patterns in the information. EK 3.2.1E

1F. Software tools, including spreadsheets and databases, help to efficiently organize and find trends in information. EK 3.2.1F

1G. Metadata is data about data. EK 3.2.1G

1H. Metadata can be descriptive data about an image, a Web page, or other complex objects. EK 3.2.1H

1I. Metadata can increase the effective use of data or data sets by providing additional information about various aspects of that data. EK 3.2.1I

2. Determine how large data sets impact the use of computational processors to discover information and knowledge. LO 3.2.2

2A. Large data sets include data such as transactions, measurements, texts, sounds, images, and videos. EK 3.2.2A

2B. The storing, processing, and curating of large data sets is challenging. EK 3.2.2B

2C. Structuring large data sets for analysis can be challenging. EK 3.2.2C

2D. Maintaining privacy of large data sets containing personal information can be challenging. EK 3.2.2D

2E. Scalability of systems is an important consideration when data sets are large. EK 3.2.2E

2F. The size or scale of a system that stores data affects how that data set is used. EK 3.2.2F

2G. The effective use of large data sets requires computational solutions. EK 3.2.2G

2H. Analytical techniques to store, manage, transmit, and process data sets change as the size of data sets scale. EK 3.2.2H

3. There are trade-offs when representing information as digital data. EU 3.3
    1. Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information. LO 3.3.1
        - 1A. Digital data representations involve trade-offs related to storage, security, and privacy concerns. EK 3.3.1A
        - 1B. Security concerns engender trade-offs in storing and transmitting information. EK 3.3.1B
        - 1C. There are trade-offs in using lossy and lossless compression techniques for storing and transmitting data. EK 3.3.1C
        - 1D. Lossless data compression reduces the number of bits stored or transmitted but allows complete reconstruction of the original data. EK 3.3.1D
        - 1E. Lossy data compression can significantly reduce the number of bits stored or transmitted at the cost of being able to reconstruct only an approximation of the original data. EK 3.3.1E
        - 1F. Security and privacy concerns arise with data containing personal information. EK 3.3.1F
        - 1G. Data is stored in many formats depending on its characteristics (e.g., size and intended use). EK 3.3.1G
        - 1H. The choice of storage media affects both the methods and costs of manipulating the data it contains. EK 3.3.1H
        - 1I. Reading data and updating data have different storage requirements. EK 3.3.1I

BI4. Algorithms BI4

1. Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages. EU 4.1

    1. Develop an algorithm for implementation in a program. LO 4.1.1

        1A. Sequencing, selection, and iteration are building blocks of algorithms. EK 4.1.1A

        1B. Sequencing is the application of each step of an algorithm in the order in which the statements are given. EK 4.1.1B

        1C. Selection uses a Boolean condition to determine which of two parts of an algorithm is used. EK 4.1.1C

        1D. Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times. EK 4.1.1D

        1E. Algorithms can be combined to make new algorithms. EK 4.1.1E

        1F. Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct. EK 4.1.1F

        1G. Knowledge of standard algorithms can help in constructing new algorithms. EK 4.1.1G

        1H. Different algorithms can be developed to solve the same problem. EK 4.1.1H

        1I. Developing a new algorithm to solve a problem can yield insight into the problem. EK 4.1.1I

    2. Express an algorithm in a language. LO 4.1.2

        2A. Languages for algorithms include natural language, pseudocode, and visual and textual programming languages. EK 4.1.2A

        2B. Natural language and pseudocode describe algorithms so that humans can understand them. EK 4.1.2B

        2C. Algorithms described in programming languages can be executed on a computer. EK 4.1.2C

        2D. Different languages are better suited for expressing different algorithms. EK 4.1.2D

        2E. Some programming languages are designed for specific domains and are better for expressing algorithms in those domains. EK 4.1.2E

        2F. The language used to express an algorithm can affect characteristics such as clarity or readability but not whether an algorithmic solution exists. EK 4.1.2F

        2G. Every algorithm can be constructed using only sequencing, selection, and iteration. EK 4.1.2G

        2H. Nearly all programming languages are equivalent in terms of being able to express any algorithm. EK 4.1.2H

2I. Clarity and readability are important considerations when expressing an algorithm in a language. EK 4.1.2I

2. Algorithms can solve many, but not all, computational problems. EU 4.2

1. Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time. LO 4.2.1

1A. Many problems can be solved in a reasonable time. EK 4.2.1A

1B. Reasonable time means that the number of steps the algorithm takes is less than or equal to a polynomial function (constant, linear, square, cube, etc.) of the size of the input. EK 4.2.1B

1C. Some problems cannot be solved in a reasonable time, even for small input sizes. EK 4.2.1C

1D. Some problems can be solved but not in a reasonable time. In these cases, heuristic approaches may be helpful to find solutions in reasonable time. EK 4.2.1D

2. Explain the difference between solvable and unsolvable problems in computer science. LO 4.2.2

2A. A heuristic is a technique that may allow us to find an approximate solution when typical methods fail to find an exact solution. EK 4.2.2A

2B. Heuristics may be helpful for finding an approximate solution more quickly when exact methods are too slow. EK 4.2.2B

2C. Some optimization problems such as "find the best" or "find the smallest" cannot be solved in a reasonable time but approximations to the optimal solution can. EK 4.2.2C

2D. Some problems cannot be solved using any algorithm. EK 4.2.2D

3. Explain the existence of undecidable problems in computer science. LO 4.2.3

3A. An undecidable problem may have instances that have an algorithmic solution, but there is no algorithmic solution that solves all instances of the problem. EK 4.2.3A

3B. A decidable problem is one in which an algorithm can be constructed to answer "yes" or "no" for all inputs (e.g., "Is the number even?"). EK 4.2.3B

3C. An undecidable problem is one in which no algorithm can be constructed that always leads to a correct yes-or-no answer. EK 4.2.3C

4. Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity. LO 4.2.4

4A. Determining an algorithm's efficiency is done by reasoning formally or mathematically about the algorithm. EK 4.2.4A

4B. Empirical analysis of an algorithm is done by implementing the algorithm and running it on different inputs. EK 4.2.4B

4C. The correctness of an algorithm is determined by reasoning formally or mathematically about the algorithm, not by testing an implementation of the algorithm. EK 4.2.4C

4D. Different correct algorithms for the same problem can have different efficiencies. EK 4.2.4D

4E. Sometimes, more efficient algorithms are more complex. EK 4.2.4E

4F. Finding an efficient algorithm for a problem can help solve larger instances of the problem. EK 4.2.4F

4G. Efficiency includes both execution time and memory usage. EK 4.2.4G

4H. Linear search can be used when searching for an item in any list; binary search can be used only when the list is sorted. EK 4.2.4H

BI5. Programming BI5

1. Programs can be developed for creative expression, to satisfy personal curiosity, to create new knowledge, or to solve problems (to help people, organizations, or society). EU 5.1

    1. Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge. LO 5.1.1

        1A. Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer. EK 5.1.1A

        1B. Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs. EK 5.1.1B

        1C. Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may be developed with different standards or methods than programs developed for widespread distribution. EK 5.1.1C

        1D. Additional desired outcomes may be realized independently of the original purpose of the program. EK 5.1.1D

        1E. A computer program or the results of running a program may be rapidly shared with a large number of users and can have widespread impact on individuals, organizations, and society. EK 5.1.1E

        1F. Advances in computing have generated and increased creativity in other fields. EK 5.1.1F

    2. Develop a correct program to solve problems. LO 5.1.2

        2A. An iterative process of program development helps in developing a correct program to solve problems. EK 5.1.2A

        2B. Developing correct program components and then combining them helps in creating correct programs. EK 5.1.2B

        2C. Incrementally adding tested program segments to correct working programs helps create large correct programs. EK 5.1.2C

        2D. Program documentation helps programmers develop and maintain correct programs to efficiently solve problems. EK 5.1.2D

        2E. Documentation about program components, such as code segments and procedures, helps in developing and maintaining programs. EK 5.1.2E

        2F. Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments. EK 5.1.2F

        2G. Program development includes identifying programmer and user concerns that affect the solution to problems. EK 5.1.2G

        2H. Consultation and communication with program users is an important aspect of program development to solve problems. EK 5.1.2H

2I. A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem. EK 5.1.2I

2J. A programmer designs, implements, tests, debugs, and maintains programs when solving problems. EK 5.1.2J

3. Collaborate to develop a program. LO 5.1.3

3A. Collaboration can decrease the size and complexity of tasks required of individual programmers. EK 5.1.3A

3B. Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming. EK 5.1.3B

3C. Collaboration in the iterative development of a program requires different skills than developing a program alone. EK 5.1.3C

3D. Collaboration can make it easier to find and correct errors when developing programs. EK 5.1.3D

3E. Collaboration facilitates developing program components independently. EK 5.1.3E

3F. Effective communication between participants is required for successful collaboration when developing programs. EK 5.1.3F

2. People write programs to execute algorithms. EU 5.2

1. Explain how programs implement algorithms. LO 5.2.1

1B. Algorithms are implemented using program instructions that are processed during program execution. EK 5.2.1B

1C. Program instructions may involved variables that are initialized and updated, read, and written. EK 5.2.1C

1D. An understanding of instruction processing and program execution is useful for programming. EK 5.2.1D

1E. Program execution automates processes. EK 5.2.1E

1F. Processes use memory, a central processing unit (CPU), and input and output. EK 5.2.1F

1G. A process may execute by itself or with other processes. EK 5.2.1G

1H. A process may execute on one or several CPUs. EK 5.2.1H

1I. Executable programs increase the scale of problems that can be addressed. EK 5.2.1I

1J. Simple algorithms can solve a large set of problems when automated. EK 5.2.1J

1K. Improvements in algorithms, hardware, and software increase the kinds of problems and the size of problems solvable by programming. EK 5.2.1K

1. Use abstraction to manage complexity in programs. LO 5.3.1

1A. Procedures are reusable programming abstractions. EK 5.3.1A

1B. A procedure is a named grouping of programming instructions. EK 5.3.1B

1C. Procedures reduce the complexity of writing and maintaining programs. EK 5.3.1C

1D. Procedures have names and may have parameters and return values. EK 5.3.1D

1E. Parameterization can generalize a specific solution. EK 5.3.1E

1F. Parameters generalize a solution by allowing a procedure to be used instead of duplicated code. EK 5.3.1F

1G. Parameters provide different values as input to procedures when they are called in a program. EK 5.3.1G

1H. Data abstraction provides a means of separating behavior from implementation. EK 5.3.1H

1I. Strings and string operations, including concatenation and some form of substring, are common in many programs. EK 5.3.1I

1J. Integers and floating-point numbers are used in programs without requiring understanding of how they are implemented. EK 5.3.1J

1K. Lists and list operations, such as add, remove, and search, are common in many programs. EK 5.3.1K

1L. Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain. EK 5.3.1L

1M. Application program interfaces (APIs) and libraries simplify complex programming tasks. EK 5.3.1M

1N. Documentation for an API/library is an important aspect of programming. EK 5.3.1N

1O. APIs connect software components, allowing them to communicate. EK 5.3.1O

1. Evaluate the correctness of a program. LO 5.4.1

1. Employ appropriate mathematical and logical concepts in programming. LO 5.5.1

BI6. The Internet BI6

1. Explain the abstractions in the Internet and how the Internet functions. LO 6.1.1

1. Explain characteristics of the Internet and the systems built on it. LO 6.2.1

2. Explain how the characteristics of the Internet influence the systems built on it. LO 6.2.2

1. Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it. LO 6.3.1

BI7. Global Impact BI7

1. Explain how computing innovations affect communication, interaction, and cognition. LO 7.1.1
2. Explain how people participate in a problem-solving process that scales. LO 7.1.2

1. Explain how computing has impacted innovations in other fields. LO 7.2.1

1. Analyze the beneficial and harmful effects of computing. LO 7.3.1

1. Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts. LO 7.4.1

1. Access, manage, and attribute information using effective strategies. LO 7.5.1
2. Evaluate online and print sources for appropriateness and credibility. LO 7.5.2