Department of Education and Early Development

# Alaska Computer Science Standards

Adopted June 2019

For additional information on Alaska's standards, write:
Standards, Department of Education & Early Development
PO Box 110500 Juneau, Alaska 99811-0500
Or call, (907) 465-2900; or visit our website: http://education.alaska.gov

## Introduction:

There is an urgent need to improve the level of public understanding of computer science as an academic and professional field. To successfully function in society, every citizen in the 21st century must understand, at the minimum, the principles of computer science. A commitment to implementing K–12 computer science standards in Alaska will aid in creating this broad public understanding and also help meet the evolving needs of the Alaskan workforce.

Computer science is an established discipline at the collegiate and post-graduate levels. It is best defined as “the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society.” All students need to understand a world that is increasingly influenced by technology and to apply computing as a tool for learning and expression across a variety of disciplines and interests.

The Alaska Computer Science Standards are meant to establish a baseline literacy in computer science for all Alaskan students and provide guidance for designing curriculum, assessments, and teacher preparation programs. The standards represent a vision in which all students, from a young age, engage in the concepts and practices of computer science. From kindergarten through 12th grade, students will develop new approaches to problem solving that harness the power of computational thinking, while not only becoming users, but creators of computing technology.

The Alaska Computer Science Standards are based on the Computer Science Teachers Association’s Interim K–12 Computer Science Standards. The standards introduce the principles and methodologies of computer science to all students, whether they are college bound or career bound after high school. The standards outlined in this document address the entire K–12 range.

K-12 educators from around the state, institutions of higher education, as well as industry were engaged as part of the standards creation process. Students proficient in Alaska’s Computer Science Standards will be prepared for either entry into postsecondary education or into positions in industry. Implementation of these standards will help ensure that Alaskans are prepared for the high-demand, good-wage jobs available across the employment landscape of Alaska and the world.

These standards do not tell teachers how to teach, nor do they attempt to override the unique qualities of each student and classroom. They simply establish a strong foundation of knowledge and skills that students need for success after graduation. It is up to schools and teachers to decide how to put the standards into practice and incorporate other state and local standards, including cultural standards.

A glossary of computer science terms is located in the Appendix where it was felt by the Standards Committee that definitions would be useful to educators.

**Legend for Identifiers**

Unique Numbering System for the Alaska Computer science K-12 Learning Standards. To help understand the organization of the standards for each individual standard here is an example.

*K.CS.D.01 = kindergarten, computing systems, devices, item 1*

| Level | Core Concept | Sub- Concepts | Number | Computer Science K–12 Learning Standard |
|---|---|---|---|---|
| Grade **K** | Computing Systems | Devises | **01** | With guidance, follow directions and make choices while using computing devices to perform a variety of tasks. |

The Legend below will help interpret the identifier for each Computer Science K-12 Learning Standard:

The identifier code corresponds to:

**Level • Concept • Sub-concepts • Number**

| **Identifier Code Level** | **Key** |
|---|---|
| K | Kindergarten |
| 1 | Grades 1 |
| Repeat 2-7 | Repeat Grades 2-7 |
| 8 | Grade 8 |
| L1 | Grades HS entry level employment competence |
| L2 | Grades HS post-secondary education |
| **Identifier Code Concepts** | **Key Concepts • Sub-concepts** |
| CS | **Computing Systems** • Devices • Hardware and Software • Troubleshooting |
| NI | **Networks and the Internet** • Network Communication and Organization • Cybersecurity |
| DA | **Data and Analysis** • Storage • Collection, Visualization and Transformation • Inference and Models |
| AP | **Algorithms and Programming** • Algorithms • Variables • Control • Modularity • Program Development |

| Identifier Code Concepts | Key Concepts • Sub-concepts |
|---|---|
| CGEI | **Community, Global and Ethical Impact**<br>• Culture<br>• Social Interactions<br>• Safety, Law and Ethics |

## Computing Systems

### Devices

- **K.CS.D.01** With guidance, follow directions and make choices while using computing devices to perform a variety of tasks.
- **1.CS.D.01** With guidance, select and use a computing device to perform a variety of tasks for an intended outcome.
- **2. CS.D.01** Select and use a computing device to perform a variety of tasks for an intended outcome.
- **3.CS.D.01** Define how computer hardware and software work together as a system to accomplish tasks (e.g., input, output, processor, sensors, and storage).
- **4.CS.D.01** Define and discuss how computer hardware and software work together as a system to accomplish tasks (e.g., input, output, processor, sensors, and storage).
- **5.CS.D.01** Define, discuss, and model how computer hardware and software work together as a system to accomplish tasks (e.g., input, output, processor, sensors, and storage).
- **6.CS.D.01** Review and analyze device(s) based on personal use and recommend improvements to the device.
- **7.CS.D.01** Review, analyze, and evaluate device(s) and how other users interact with devices and recommend improvements to design.
- **8.CS.D.01** Develop and implement a process to evaluate existing computing devices and recommend improvements to design based on analysis of how other users interact with the device.
- L1.CS.D.01 Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.

### Hardware and Software

- K.CS.HS.01 Use accurate terminology to locate and identify common computing devices and components, in a variety of environments (e.g., laptop, tablet, mouse).
- **1.CS.HS.01** Use accurate terminology in naming and describing the function of common computing devices and components (e.g., laptop, tablet, mouse).
- **2.CS.HS.01** Identify computer system components and peripherals, and their basic use (e.g., hard drive, memory, processor).
- **3.CS.HS.01** Define binary, and how that relates to computers transmitting information. Review Binary's relationship to off and on.
- **4.CS.HS.01** Define and discuss how information flows through hardware and software to accomplish tasks using binary.

- **5.CS.HS.01** Define, discuss, and model how information flows through hardware and software to accomplish tasks such as converting a word to binary.
- **6.CS.HS.01** Identify ways that hardware and software (User Interface) design are combined to collect and exchange data.
- **7.CS.HS.01** Evaluate and recommend improvements to hardware and software (User Interface) design to collect and exchange data.
- **8.CS.HS.01** Design and refine projects that combine hardware and software (User Interface) to collect and exchange data. L1.CS.HS.01 Explain the interactions between application software, system software, and hardware.
- **L2.CS.HS.01** Identify the interactions of an operating system between software and hardware.

## Troubleshooting

- **K.CS.T.01** Recognize that computing systems might not work as expected and use accurate terminology to identify simple hardware or software problems (e.g., volume turned down on headphones, monitor turned off).
- **1.CS.T.01** Identify simple hardware and software problems that may occur during use (e.g., app or program is not working as expected; no sound is coming from the device; caps lock turned on).
- **2.CS.T.01** Identify simple hardware and software problems that may occur during use and discuss problems with peers and adults (e.g., app or program is not working as expected; no sound is coming from the device; caps lock turned on).
- **3.CS.T.01** Identify and discuss simple hardware and software problems that may occur during everyday use.
- **4.CS.T.01** Identify and research simple hardware and software problems that may occur during everyday use.
- **5.CS.T.01** Identify, discuss, and apply strategies for solving simple hardware and software problems that may occur using everyday use (e.g., rebooting the device; checking the power; force shut down of an application).
- **6.CS.T.01** Identify and discuss increasingly complex software and hardware problems with computing devices and their components.
- **7.CS.T.01** Identify and fix increasingly complex software and hardware problems with computing devices and their components.
- **8.CS.T.01** Systematically identify, fix, and document increasingly complex software and hardware problems with computing devices and their components.
- **L1.CS.T.01** Develop and apply criteria for systematic discovery of errors and strategies for correction in computing systems.

## Network and the Internet

### Network Communication and Organization

- **K.NI.NCO.01** Recognize that computing devices can be connected together.
- **1.NI.NCO.01** Recognize that by connecting computing devices together they can share information (e.g., remote storage, printing, the internet).
- **2.NI.NCO.01** Recognize that computing devices can be connected at various scales (e.g., Bluetooth, Wi-Fi).
- **3.NI.NCO.01** Recognize that information is sent and received over both physical or wireless paths.
- **4.NI.NCO.01** Recognize and explain how information is sent and received over both physical and wireless paths.
- **5.NI.NCO.01** Recognize, explain, and model how information is broken down into packets (smaller pieces), transmitted between devices, and reassembled.
- **6.NI.NCO.01** Model a simple protocol for transferring information using packets.
- **7.NI.NCO.01** Explain how a system recovers when a packet is lost and the effect it has on the transferred information.
- **8.NI.NCO.01** Explain protocols and their importance to data transmission; model how packets are broken down into smaller pieces and how they are delivered.
- **L1.NI.NCO.01** Evaluate the scalability and reliability of networks by identifying and illustrating the basic components of computer networks and network protocols (e.g., routers, switches, servers, IP, DNS).
- **L2.NI.NCO.01** Describe the issues that impact network functionality (e.g., bandwidth, load, delay, topology).

### Cybersecurity

- **K.NI.C.01** Discuss what passwords are and why we do not share them with others. With guidance, use passwords to access technological devices, apps, etc.
- **1.NI.C.01** Identify what passwords are; explain why they are not shared; and discuss what makes a password strong. Independently, use passwords to access technological devices, apps etc.
- **2.NI.C.01** Explain what passwords are; why we use them, and write and design strong passwords to protect devices and information from unauthorized access. Identify other forms of authentication (biometrics).
- **3.NI.C.01** Discuss basic issues related to responsible use of technology and information and describe personal consequences of inappropriate use.
- **4.NI.C.01** Define cybersecurity and discuss real-world cybersecurity problems and how physical and digital personal information can be protected.
- **5.NI.C.01** Discuss real-world cybersecurity problems and how personal information

can be protected. Discussion topics could be based on current events related to cybersecurity or topics that are applicable to students.

- **6.NI.C.01** Identify existing cybersecurity concerns with the Internet and systems it uses.
- **6.NI.C.02** Explain the importance of secured websites and describe how one method of encryption works.
- **7.NI.C.01** Explain how to protect electronic information, both physical and digital, identify cybersecurity concerns and options to address issues with the Internet and the systems it uses
- **7.NI.C.02** Identify and explain two or more methods of encryption used to ensure and secure the transmission of information.
- **8.NI.C.01** Evaluate physical and digital procedures that could be implemented to protect electronic data/information. Explain the impacts of malware (e.g., hacking, ransomware).
- **8.NI.C.02** Compare the advantages and disadvantages of multiple methods of encryption to model the secure transmission of information.
- **L1.NI.C.01** Compare various security measures by evaluating tradeoffs between the usability and security of a computing system.
- **L1.NI.C.02** Illustrate how sensitive data can be affected by attacks.
- **L1.NI.C.03** Recommend security measures to address various scenarios based on the principles of information security.
- **L1.NI.C.04** Explain tradeoffs when selecting and implementing cybersecurity recommendations from multiple perspectives such as the user, enterprise, and government.
- **L2.NI.C.01** Compare and refine ways in which software developers protect devices and information from unauthorized access.
- **L2.NI.C.02** Learn detection and prevention methods to respond to attacks on sensitive data. Develop a response plan that enables recovery from such attacks.

## Data Analysis

### Storage

- **K.DA.S.01** Identify types of data from our everyday lives and computing devices (e.g., digital images, videos, apps, documents)
- **1.DA.S.01** With guidance, locate, open, modify and save an existing file with a computing device.
- **2.DA.S.01** With guidance, create, copy, locate, modify, and delete a file on a computing device and define the information stored as data.

- **3.DA.S.01** Recognize that different applications work with different types of files (e.g., images, documents).
- **4.DA.S.01** Explain how many kilobytes make one megabyte, and how many megabytes make one gigabyte.
- **5.DA.S.01** Using correct terminology explain why various types of files differ in size (e.g., video, images and documents).
- **6.DA.S.01** Identify how the same data can be represented in multiple ways.
- **7.DA.S.01** Create multiple representations of data.
- **8.DA.S.01** Analyze multiple methods of representing data and choose the most appropriate method for representing data.
- **L1.DA.S.01** Translate and compare different bit representations of real-world phenomena, such as characters, numbers, and images.
- **L1.DA.S.02** Review different database types.
- **L2.DA.S.01** Evaluate and explain the various types of databases, with their specific benefits and limitation.

## Collection, Visualization and Transformation

- **K.DA.CVT.01** With guidance, collect data and present it visually.
- **1.DA.CVT.01** With guidance, collect data and present it two different ways.
- **2.DA.CVT.01** With guidance, collect and present the same data in various visual formats.
- **3.DA.CVT.01** Collect and organize data in a spreadsheet.
- **4.DA.CVT.01** Collect and organize data to highlight and display relationships.
- **5.DA.CVT.01** Collect, organize, interpret, and display data to highlight relationships and support a claim.
- **6.DA.CVT.01** Collect data using computational tools and transform the data to make it more useful (e.g., spreadsheet formulas)
- **7.DA.CVT.01** Collect data using computational tools and transform the data to make it more useful and reliable.
- **8.DA.CVT.01** Develop, implement, and refine a process that utilizes computational tools to collect and transform data to make it more useful and reliable.
- **L1.DA.CVT.01** Use tools and techniques to locate, collect and create visualizations of small and large-scale data sets (e.g., paper surveys, online data sets, etc.).
- **L2.DA.CVT.01** Use data analysis tools and techniques to identify patterns from complex real- world phenomena.
- **L2.DA.CVT.02** Generate data sets that support a claim or communicates information using a variety of data collection tools and analysis techniques.

### Inference and Models

- **K.DA.IM.01** With guidance, draw conclusions and make predictions based on picture graphs or patterns (e.g., make predictions based on weather data presented in a picture graph; complete a pattern).
- **1.DA.IM.01** With guidance, interpret data from a chart or graph (visualization) in order to make a prediction, with or without a computing device.
- **2.DA.IM.01** With guidance, construct and interpret data and present it in a chart or graph (visualization) in order to make a prediction, with or without a computing device.
- **3.DA.IM.01** With guidance, utilize data to make predictions and discuss whether there is adequate data to make reliable predictions.
- **4.DA.IM.01** Determine the accuracy of conclusions and how they are influenced by the amount of data collected.
- **5.DA.IM.01** Use data to highlight or propose cause and effect relationships, predict outcomes, or communicate an idea.
- **6.DA.IM.01** Use models and simulations to formulate, refine, and test hypotheses.
- **7.DA.IM.01** Discuss the correctness of a model representing a system by comparing the model's generated results with observed data from the modeled system.
- **8.DA.IM.01** Refine computational models based on the data generated by the models.
- **L1.DA.IM.01** Use computational models such as data analysis, pattern recognition, and/or simulations to show the relationships between collected data elements.
- **L2.DA.IM.01** Use models and simulations to help formulate, refine, and test scientific hypotheses.

## Algorithms and Programming

### Algorithms

- K.AP.A.01 With guidance, model daily processes and follow algorithms (sets of step-by-step instructions) to complete tasks verbally, kinesthetically, with robot devices, or a programing language.
- **1.AP.A.01** With guidance, model daily processes and follow algorithms (sets of step-by-step instructions) to complete tasks verbally, kinesthetically, with robot devices, or a programing language.
- **2.AP.A.01** With guidance, model daily processes by creating and following algorithms (sets of step-by- step instructions) to complete tasks verbally, kinesthetically, with robot devices, or a programing language.
- **3.AP.A.01** Create and follow algorithms to accomplish a simple task or solve a simple problem
- **4.AP.A.01** Create, compare & refine multiple algorithms for the same task.

- **5..AP.A.01** Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
- **6.AP.A.01** Use an existing algorithm in natural language or pseudocode to solve complex problems.
- **7.AP.A.01** Select and modify an existing algorithm in natural language or pseudocode to solve complex problems.
- **8.AP.A.01** Design algorithms in natural language, flow and control diagrams, comments within code, and/or pseudocode to solve complex problems.
- **L1.AP.A.01** Use algorithms (e.g., sequencing, selection, iteration, recursion, etc.) to create a prototype to provide a possible solution for a common problem.
- **L2.AP.A.01** Describe how artificial intelligence drives many software and physical systems (e.g., autonomous robots, computer vision, pattern recognition, text analysis).
- **L2.AP.A.02** Develop an artificial intelligence algorithm to play a game against a human opponent or solve a common problem.
- **L2.AP.A.03** Critically examine and adapt classic algorithms (e.g. selection sort, insertion sort, etc.).
- **L2.AP.A.04** Evaluate algorithms (e.g., sorting, searching) in terms of their efficiency, correctness, and clarity.

## Variables

- **K.AP.V.01** With guidance, model and represent grade level appropriate data (e.g., print, numbers, kinesthetic movement, symbols, and robot manipulatives).
- **1.AP.V.01** With guidance, model and represent grade level appropriate data (e.g., print, numbers, kinesthetic movement, symbols and robot manipulatives).
- **2.AP.V.01** Model the way a computer program manipulates grade level appropriate data (e.g., print, numbers, kinesthetic movement, symbols and robot manipulatives).
- **3.AP.V.01** Create programs that use variables to store and modify grade level appropriate data.
- **4.AP.V.01** Create programs that use variables to store and modify grade level appropriate data.
- **5.AP.V.01** Create programs that use variables to store and modify grade level appropriate data.
- **6.AP.V.01** Develop programs that utilize combinations of repetition, conditionals, functions, and the manipulation of variables representing different data types.
- **7.AP.V.01** Develop programs that utilize combinations of repetition, compound conditionals, functions, and the manipulation of variables representing different data types.

- **8.AP.V.01** Develop programs that utilize combinations of nested repetition, compound conditionals, functions, and the manipulation of variables representing different data types.
- **L1.AP.V.01** Demonstrate the use of lists to simplify solutions and to generalize computation problems instead of repeatedly using simple variables.
- **L2.AP.V.01** Compare and contrast simple data structures and their uses (e.g., arrays, lists, stacks, queues, maps, trees, graphs, and databases).

## Control

- **K.AP.C.01** With guidance, create programs to accomplish tasks as a means of creative expression using a programming language, robot device or unplugged activity, either independently or collaboratively, including sequencing, emphasizing the beginning, middle, and end.
- **1.AP.C.01** With guidance, create programs to accomplish tasks as a means of creative expression or problem solving using a programming language, robot device or unplugged activity, either independently or collaboratively including sequencing and repetition.
- **2.AP.C.01** With guidance, create programs using a programming language, robot device or unplugged activity that utilize sequencing and repetition to solve a problem or express ideas both independently and collaboratively.
- **L1.AP.C.01** Justify the selection of specific control structures when trade-offs involve implementation, readability, and program performance.
- **L1.AP.C.02** Develop an event-based program that will solve a practical problem, or allow self-expression.
- **L2.AP.C.01** Trace the execution of recursive algorithms, illustrating output and changes in values of named variables.

## Modularity

- K.AP.M.01 With guidance, decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.
- **1.AP.M.01** Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.
- **2.AP.M.01** Decompose (break down) and explain the steps needed to solve a problem into a precise sequence of instructions.
- **3.AP.M.01** Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.
- **4.AP.M.01** Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.
- **5.AP.M.01** Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.

- **6.AP.M.01** Decompose (break down) problems into abstraction layers to facilitate the design, implementation, and review of programs.
- **7.AP.M.01** Decompose (break down) problems into abstraction layers to facilitate the design, implementation, and review of increasingly complex programs.
- **8.AP.M.01** Decompose (break down) problems and sub-problems into abstraction layers to facilitate the design, implementation, and review of complex programs.
- **L1.AP.M.01** Using systematic analysis and design, break down a solution into basic elements such as procedures, functions, or methods.
- **L1.AP.M.02** Create computational artifacts by using common structures to organize, manipulate and/or process data.
- **L2.AP.M.01** Construct solutions to problems using student-created components, such as functions, procedures, modules, and/or objects.
- **L2.AP.M.02** Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution.
- **L2.AP.M.03** Create programming solutions using code reuse and applied technique with appropriate attribution (e.g., libraries, APIs, collaboration software, and versioning software).

## Program Development

- **K.AP.PD.01** With guidance, create a grade level appropriate document to illustrate thoughts, ideas, and stories in a sequential (step-by-step) manner (e.g., story map, storyboard, and sequential graphic organizer).
- **K.AP.PD.02** Independently or with guidance give credit to ideas, creations and solutions of others while writing and/or developing programs.
- **K.AP.PD.03** Independently and collaboratively, identify and correct errors in an algorithm that includes sequencing and repeated procedures using a programming language or unplugged activities.
- **K.AP.PD.04** Use correct terminology (first, second...) in the development of an algorithm to solve a simple problem
- **1.AP.PD.01** Independently or with guidance, create a grade level appropriate document to illustrate thoughts, ideas, and stories in a sequential (step-by- step) manner (e.g., story map, storyboard, and sequential graphic organizer).
- **1.AP.PD.02** Independently or with guidance give credit to ideas, creations and solutions of others while writing and/or developing programs.
- **1.AP.PD.03** Independently and collaboratively, debug programs, which include sequencing and repetition to accomplish tasks as a means of creative expression or problem solving using a programming language and/or unplugged activities.

- **1.AP.PD.04** Use correct terminology (beginning, middle, end...), and explain the choices made in the development of an algorithm and/or program to solve a simple problem.
- **2.AP.PD.01** Independently or with guidance, create a grade level appropriate document to illustrate thoughts, ideas, and stories in a sequential (step-by- step) manner (e.g., story map, storyboard, and sequential graphic organizer).
- **2.AP.PD.02** Independently or with guidance give credit to ideas, creations and solutions of others while writing and/or developing programs.
- **2.AP.PD.03** Independently and collaboratively, create and debug programs, which include sequencing and repetition to accomplish tasks as a means of creative expression or problem solving using a programming language and/or unplugged activities.
- **2.AP.PD.04** Use correct terminology (debug, program input/output, code ...) to explain the development of an algorithm to solve a problem in an unplugged activity, hands on manipulatives or a programming language.
- **3.AP.PD.01** Outline problems and potential solutions using a sequence of steps and conditional logic (e.g., if-then-else statements).
- **3.AP.PD.02** Observe intellectual property rights and give appropriate credit when creating or remixing programs.
- **4.AP.PD.01** Create and debug programs using variables, loops, functions that intake, store, and display data.
- **4.AP.PD.02** Observe intellectual property rights and give appropriate credit when creating or remixing programs.
- **5.AP.PD.01** Define the concept of abstraction and create increasingly complex programs.
- **5.AP.PD.02** Observe intellectual property rights and give appropriate credit when creating or remixing programs.
- **6.AP.PD.01** Seek and incorporate feedback from team members to refine a solution to a problem.
- **6.AP.PD.02** Incorporate existing code, media, and libraries into original programs and give attribution.
- **6.AP.PD.03** Test and refine programs using teacher provided inputs.
- **6.AP.PD.04** Break down tasks and follow an individual timeline when developing a computational artifact.
- **6.AP.PD.05** Document block-based or text-based programs in order to make them easier to follow, test, and debug.
- **7.AP.PD.01** Seek and incorporate feedback from team members and users to refine a solution to a problem.
- **7.AP.PD.02** Incorporate existing code, media, and libraries into original programs of increasing complexity and give attribution.

- **7.AP.PD.03** Test and refine programs using a variety of student created inputs.
- **7.AP.PD.04** Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
- 7.AP.PD.05 Document block-based or text-based programs of increasing complexity in order to make them easier to follow, test, and debug.
- 8.AP.PD.01 Seek and incorporate feedback from team members and users to refine a solution to a problem that meets the needs of diverse users.
- **8.AP.PD.02** Incorporate existing code, media, and libraries into original programs of increasing complexity and give attribution.
- **8.AP.PD.03** Systematically test and refine programs using a range of student created inputs.
- **8.AP.PD.04** Explain how effective communication between participants is required for successful collaboration when developing computational artifacts.
- **L1.AP.PD.01** Create software by analyzing a problem and/or process, developing a solution, testing outcomes, debug, documenting, and adapting the program for a variety of users.
- **L1.AP.PD.02** Classify and define a variety of software licensing schemes and discuss the advantages and disadvantages of the different schemes in software development (e.g. open source, freeware, commercial).
- **L1.AP.PD.03** Evaluate and refine computational artifacts to make them more user-friendly, efficient and/or accessible.
- **L1.AP.PD.04** Design and develop a computational artifact while working in a team.
- **L1.AP.PD.05** Using visual aids and documentation, illustrate the design elements and data flow of the development of a complex program (e.g. flowcharts, pseudocode, etc.).
- **L2.AP.PD.01** Compare multiple programming languages and discuss features that make them useful for solving problems and developing systems.
- **L2.AP.PD.02** Using the software life cycle process, create software that will provide solutions for a variety of users.
- **L2.AP.PD.03** Design software in a project team environment using Agile Development methods (e.g., versioning and collaboration systems).
- **L2.AP.PD.04** Explain security issues that might lead to compromised computer programs.
- **L2.AP.PD.05** Develop programs for multiple computing platforms.
- **L2.AP.PD.06** Develop and use a series of test cases to verify that a program performs according to its design specifications.
- **L2.AP.PD.07** Through peer review systematically check code for correctness, usability, readability, efficiency, portability, and scalability (e.g. code review).

- L2.AP.PD.08 Modify an existing program to add additional functionality and discuss intended and unintended implications with appropriate attribution.

## Community, Global and Ethical Impacts

### Culture

- **K.CGEI.C.01** List different ways in which computing devices are used in your daily life.
- **1.CGEI.C.01** Identify how people use many types of technologies in their daily work and personal lives.
- **2.CGEI.C.01** List different technology tools, and describe how people use them in their daily work and personal lives.
- **3.CGEI.C.01** Discuss ongoing trends in technologies that have changed the world, and express how those trends influence and are influenced by cultural practices.
- **3.CGEI.C.02** Brainstorm ways to improve the accessibility and usability of technology product for the diverse needs and wants of users.
- **4.CGEI.C.01** Discuss ongoing trends in technologies that have changed the world, and express how those trends influence and are influenced by cultural practices.
- **4.CGEI.C.02** Brainstorm ways to improve the accessibility and usability of technology pro ducts for the diverse needs and wants of users.
- **5.CGEI.C.01** Discuss ongoing trends in technologies that have changed the world, and express how those trends influence and are influenced by cultural practices.
- **5.CGEI.C.02** Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users.
- **6.GCEI.C.01** Explain how computing impacts people's' everyday activities and explore carriers related to the field of computer science.
- **6.GCEI.C.02** Identify and discuss the technology proficiencies needed in the classroom and the workplace, and how to meet the needs of diverse users.
- **7.GCEI.C.01** Explain how computing impacts innovation in other fields and explore carriers related to the field of computer science.
- **7.GCEI.C.02** Relate the distribution of computing resources in a global society to issues of equity, access, and power.
- **8.GCEI.C.01** Describe the trade-offs associated with computing technologies, explaining their effects on economies and global societies, and explore community and global careers related to the field of computer science (e.g., automation).
- **8.GCEI.C.02** Evaluate and improve the design of existing technologies to meet the needs of diverse users and increase accessibility and usability. Evaluate how technology can be used to distort, exaggerate, and misrepresent information.
- **L1.CGEI.C.01** Test and refine computational artifacts to reduce bias and equity deficits.

- **L1.CGEI.C.02** Demonstrate how a given algorithm applies to problems across disciplines.
- **L2.CGEI.C.01** Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society.
- **L2.CGEI.C.02** Based on research, evaluate how computing has revolutionized an aspect of our culture and predict how it might evolve (e.g., education, healthcare, art/entertainment, and energy).

## Social Interactions

- **K.CGEI.SI.01** With guidance, identify appropriate manners while participating in a digital community.
- **1.CGEI.SI.01** With guidance, identify appropriate and inappropriate behavior. Act responsibly while participating in an online community and know how and who to report concerns.
- **2.CGEI.SI.01** Develop a code of conduct, explain, and practice grade-level appropriate behavior and responsibilities while participating in a digital community. Identify and report inappropriate behavior.
- **3.CGEI.SI.01** Develop a code of conduct, explain, and practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.
- **3.CGEI.SI.02** Identify how computational products may be, or have been, improved to incorporate diverse perspectives. Seek diverse perspectives for the purpose of improving computational artifacts.
- **4.CGEI.SI.01** Develop a code of conduct, explain, and practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.
- **4.CGEI.SI.02** As a collaborative team, consider and leverage each other's diverse perspectives on improving a computational product.
- **5.CGEI.SI.01** Develop a code of conduct, explain, and practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.
- **5.CGEI.SI.02** As a team, collaborate with outside resources (other grade levels, online collaborative spaces) to include diverse perspectives to improve computational products.
- **6.GCEI.SI.01** Individually and collaboratively develop and conduct an online survey that seeks input from a broad audience. Describe and use safe, appropriate, and responsible practices (netiquette) when participating in online communities (e.g., discussion groups, blogs, social networking sites).
- **7.GCEI.SI.01** Individually and collaboratively use advanced tools to design and create online content (e.g., digital portfolio, multimedia, blog, web page). Describe and use safe, appropriate, and responsible practices (netiquette) when participating in online communities (e.g., discussion groups, blogs, social networking sites).

- **8.GCEI.SI.01** Communicate and publish key ideas and details individually or collaboratively in a way that informs, persuades, and/or entertains using a variety of digital tools and media-rich resources. Describe and use safe, appropriate, and responsible practices (netiquette) when participating in online communities (e.g., discussion groups, blogs, social networking sites).
- **L1.CGEI.SI.01** Demonstrate how computing increases connectivity to people in various cultures.

## Safety, Law and Ethics

- **K.CGEI.SLE.01** With guidance, keep log in information private, and log off of devices appropriately
- **1.CGEI.SLE.01** Keep log in information private, and log off of devices appropriately.
- **2.CGEI.SLE.01** Be able to explain the reasoning of keeping login information private. Keep log in information private, and log off of devices appropriately.
- **3.CGEI.SLE.01** Identify types of digital data that may have intellectual property rights that prevent copying or require attribution.
- **4.CGEI.SLE.01** Discuss the social impact of violating intellectual property rights. Use public domain or creative commons media, and refrain from copying or using material created by others without permission.
- **5.CGEI.SLE.01** Observe intellectual property rights, give appropriate credit when using resources and consider the licenses on computational artifacts while using resources.
- **6. GCEI.SLE.01** Differentiate between appropriate and inappropriate content on the Internet, and identify unethical and illegal online behavior.
- **6.GCEI.SLE.02** Identify what distinguishes humans from machines focusing on human intelligence versus machine intelligence (e.g., robot motion; speech and language understanding; computer vision)
- **7. GCEI.SLE.01** Explain the connection between the longevity of data on the Internet, personal online identity, and personal privacy.
- **7.GCEI.SLE.02** Describe ways in which computers use models of intelligent behavior (e.g., robot motion; speech and language understanding; computer vision)
- **8. GCEI.SLE.01** Discuss the social impacts and ethical considerations associated with cybersecurity, including the positive and malicious purposes of hacking.
- **8.GCEI.SLE.02** Compare and contrast human intelligence and computer intelligence (e.g., emotional decision making versus logical decisions; common sense; literal versus abstract).
- **L1.CGEI.SLE.1** Explain the beneficial and harmful effects that intellectual property laws can have on innovation.

- **L1.CGEI.SLE.2** Explain the privacy concerns related to the collection and large scale analysis of information about individuals that may not be evident to users (e.g., how businesses, social media, and the government collects and uses data).
- **L1.CGEI.SLE.3** Evaluate the social and economic implications as related to privacy, data, property, information, and identity in the context of safety, law, or ethics.
- **L1.CGEI.SLE.4** Describe the beneficial and intrusive aspects of advancing and emerging technologies (e.g., artificial intelligent agents, IoT, robotics).
- **L1.CGEI.SLE.5** Discuss diverse careers that are influenced by computer science and its availability to all regardless of background.
- **L2.CGEI.SLE.1** Debate laws and regulations that impact the development and use of software.
- **L2.CGEI.SLE.2** Identify the ethical and moral implications encountered in managing and curating knowledge (e.g., harvesting; information overload; knowledge management; reposting; sharing; summarizing).
- **L2.CGEI.SLE.3** Explain how cutting-edge technology may affect the way business is conducted in the future (e.g., eCommerce, entrepreneurship, payment methods, business responsibilities).

## Appendix A:

# Glossary

### A

**Abstraction:** The process of reducing complexity by focusing on the main idea. By hiding details irrelevant to the question at hand and bringing together related and useful details, abstraction reduces complexity and allows one to focus on the problem.

**Accessibility:** The design of products, devices, services, or environments for people who experience disabilities. Accessibility standards that are generally accepted by professional groups include the Web Content Accessibility Guidelines (WCAG) 2.0 and Accessible Rich Internet Applications (ARIA) standards.

**Algorithm:** A step-by-step process to complete a task.

**App**: A type of application software designed to run on a mobile device, such as a smartphone or tablet computer. Also known as a mobile application.

**Artifact**: Anything created by a human. See computational artifact for the definition used in computer science.

**Artificial Intelligence (A.I.)**: The branch of computer science dealing with the simulation of intelligent behavior in computers.

**Audience:** Expected end users of a computational artifact or system.

**Authentication:** The verification of the identity of a person or process.

### B

**Binary**: A numeric system that only uses two digits — 0 and 1. Computers operate in binary, meaning they store data and perform calculations using only zeros and ones.

**Block-based**: Using pre-programmed "blocks" of instructions to teach kids coding.

### C

**Code**: Any set of instructions expressed in a programming language.

**Complexity:** The minimum amount of resources, such as memory, time, or messages, needed to solve a problem or execute an algorithm.

**Component:** An element of a larger group. Usually, a component provides a particular service or group of related services. Computational: Relating to computers or computing methods.

**Computational artifact**: Anything created by a human using a computational thinking process and a computing device. A computational artifact can be, but is not limited to, a program, image, audio, video, presentation, or web page file.

**Computational thinking**: The human ability to formulate problems so that their solutions can be represented as computational steps or algorithms to be executed by a computer.

**Computer:** A machine or device that performs processes, calculations, and operations based on instructions provided by a software or hardware program.

**Computer science**: The study of computers and algorithmic processes, including their principles, their hardware and software designs, their implementation, and their impact on society.

**Computing**: Any goal-oriented activity requiring, benefiting from, or creating algorithmic processes.

**Computing device**: A physical device that uses hardware and software to receive, process, and output information. Computers, mobile phones, and computer chips inside appliances are all examples of computing devices.

**Computing system**: A collection of one or more computers or computing devices, together with their hardware and software, integrated for the purpose of accomplishing shared tasks. Although a computing system can be limited to a single computer or computing device, it more commonly refers to a collection of multiple connected computers, computing devices, and hardware.

**Conditional**: A feature of a programming language that performs different computations or actions depending on whether a programmer-specified Boolean condition evaluates to true or false.

**Configuration:**

- (process): Defining the options that are provided when installing or modifying hardware and software or the process of creating the configuration (product).
- (product): The specific hardware and software details that tell exactly what the system is made up of, especially in terms of devices attached, capacity, or capability.

**Connection:** A physical or wireless attachment between multiple computing systems, computers, or computing devices. Connectivity: A program's or device's ability to link with other programs and devices.

**Control structure:** A programming (code) structure that implements control. Conditionals and loops are examples of control structures.

**Culture:** A human institution manifested in the learned behavior of people, including their specific belief systems, language(s), social relations, technologies, institutions, organizations, and systems for using and developing resources.

**Cybersecurity:** The protection against access to, or alteration of, computing resources through the use of technology, processes, and training.

## D

**Data:** Information that is collected and used for reference or analysis. Data can be digital or nondigital and can be in many forms, including numbers, text, show of hands, images, sounds, or video.

**Data structure**: A particular way to store and organize data within a computer program to suit a specific purpose so that it can be accessed and worked with in appropriate ways.

**Data type**: A classification of data that is distinguished by its attributes and the types of operations that can be performed on it. Some common data types are integer, string, Boolean (true or false), and floating-point.

**Debug/debugging**: The process of finding and correcting errors (bugs) in programs.

**Decompose/decomposition**: Breaking down a problem or system into components.

**Device:** A unit of physical hardware that provides one or more computing functions within a computing system. It can provide input to the computer, accept output, or both.

**Digital citizenship**: The norms of appropriate, responsible behavior with regard to the use of technology.

## E

**Efficiency:** A measure of the amount of resources an algorithm uses to find an answer. It is usually expressed in terms of the theoretical computations, the memory used, the number of messages passed, the number of disk accesses, etc.

**Encryption:** The conversion of electronic data into another form, called ciphertext, which cannot be easily understood by anyone except authorized parties.

**Event:** Any identifiable occurrence that has significance for system hardware or software. User-generated events include keystrokes and mouse clicks; system-generated events include program loading and errors.

**Execute:** To carry out (or "run") an instruction or set of instructions (program, app, etc.).

## G

**Gigabyte**: A unit of computer memory. One gigabyte equals 1,000,000,000 bytes.

## H

**Hacking:** Gaining unauthorized access to other computers.

**Hardware**: The physical components that make up a computing system, computer, or computing device.

## I

**Implementation**: The process of expressing the design of a solution in a programming language (code) that can be made to run on a computing device.

**Inference:** A conclusion reached on the basis of evidence and reasoning. Input: The signals or instructions sent to a computer.

**Internet**: The global collection of computer networks and their connections, all using shared protocols to communicate.

**Internet of Things** (IoT): An umbrella term that refers to anything connected to the Internet. It includes traditional computing devices, but also includes a growing list of other devices, including home appliances, automobiles, wearable electronics and security cameras.

**Iterative**: Involving the repeating of a process with the aim of approaching a desired goal, target, or result.

## K

**Kilobyte**: A unit of computer memory. One kilobyte equals 1,000 bytes.

## L

**Loop:** A programming structure that repeats a sequence of instructions as long as a specific condition is true.

## M

**Megabyte:** A unit of computer memory. One megabyte equals 1,000,000 bytes. Memory: Temporary storage used by computing devices.

**Model**: A representation of some part of a problem or a system. Note: This definition differs from that used in science.

**Modularity:** The characteristic of a software/web application that has been divided (decomposed) into smaller modules. An application might have several procedures that are called from inside its main procedure. Existing procedures could be reused by recombining them in a new application.

**Module**: A software component or part of a program that contains one or more procedures. One or more independently developed modules make up a program.

## N

**Network**: A group of computing devices (personal computers, phones, servers, switches, routers, etc.) connected by cables or wireless media for the exchange of information and resources.

## O

**Operation**: An action, resulting from a single instruction that changes the state of data.

## P

**Packet**: The unit of data sent over a network.

**Procedure**: An independent code module that fulfills some concrete task and is referenced within a larger body of program code. The fundamental role of a procedure is to offer a single point of reference for some small goal or task that the developer or programmer can trigger by invoking the procedure itself.

**Process:** A series of actions or steps taken to achieve a particular outcome.

**Program:** A set of instructions that the computer executes to achieve a particular objective.

**Programming**: The craft of analyzing problems and designing, writing, testing, and maintaining programs to solve them.

**Protocol**: The special set of rules used by endpoints in a telecommunication connection when they communicate. Protocols specify interactions between the communicating entities.

## R

**Reliability**: An attribute of any system that consistently produces the same results, preferably meeting or exceeding its requirements.

**Remix:** The process of creating something new from something old. Originally a process that involved music, remixing involves creating a new version of a program by recombining and modifying parts of existing programs, and often adding new pieces, to form new solutions.

**Router**: A device or software that determines the path that data packets travel from source to destination.

## S

**Scalability**: The capability of a network to handle a growing amount of work or its potential to be enlarged to accommodate that growth. [Wikipedia]

**Software:** Programs that run on a computing system, computer, or other computing device.

**Storage**:

- (place) A place, usually a device, into which data can be entered, in which the data can be held, and from which the data can be retrieved at a later time.
- (process) A process through which digital data is saved within a data storage device by means of computing technology. Storage is a mechanism that enables a computer to retain data, either temporarily or permanently.

**String:** A sequence of letters, numbers, and/or other symbols. A string might represent, for example, a name, address, or song title. Some functions commonly associated with strings are length, concatenation, and substring.

**Structure:** A general term used in the framework to discuss the concept of encapsulation without specifying a particular programming methodology.

**Switch:** A high-speed device that receives incoming data packets and redirects them to their destination on a local area network (LAN).

**System:** A collection of elements or components that work together for a common purpose.

## T

**Test case:** A set of conditions or variables under which a tester will determine whether the system being tested satisfies requirements or works correctly.

**Topology**: The physical and logical configuration of a network; the arrangement of a network, including its nodes and connecting links. A logical topology is the way devices appear connected to the user. A physical topology is the way they are actually interconnected with wires and cables.

**Troubleshooting**: A systematic approach to problem solving that is often used to find and resolve a problem, error, or fault within software or a computing system.

## U

**User Interface (U.I.)**: The means in which a person controls a software application or hardware device.

## V

**Variable:** A symbolic name that is used to keep track of a value that can change while a program is running. Variables are not just used for numbers; they can also hold text, including whole sentences (strings) or logical values (true or false). A variable has a data type and is associated with a data storage location; its value is normally changed during the course of program execution. Note: This definition differs from that used in math.