Arizona Computer Science Standards

Kindergarten – Second Grade

ARIZONA DEPARTMENT OF EDUCATION
HIGH ACADEMIC STANDARDS FOR STUDENTS
Adopted October 2018

Contents

Vision Statement

Arizona's K-12 students will develop a foundation of computer science knowledge and learn new approaches to problem solving and critical thinking. Students will become innovative, collaborative creators and ethical, responsible users of computing technology to ensure they have the knowledge and skills to productively participate in a global society.

Introduction

Understanding problems, their potential solutions, and the technologies, techniques, and resources needed to solve them are vital for citizens of the 21st century. The State of Arizona has created computer science standards to further this understanding. These standards allow students to develop a foundation of computer science knowledge. By learning new approaches to problem solving that capture the power of computational thinking, students become users and creators of computing technology. The computer science standards will empower students to:

- Be informed citizens who can critically engage in public discussion on computer science related topics
- Develop as learners, users, and creators of computer science knowledge and artifacts
- Understand the role of computing in the world around them
- Learn, perform, and express themselves critically in a variety of subjects and interests

As the foundation for all computing, computer science is "the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society" (Tucker et. al, 2006, p. 2). Computer science builds upon the concepts of computer literacy, educational technology, digital citizenship, and information technology. The previously listed concepts tend to focus more on using computer technologies as opposed to understanding why they work and how to create those technologies (K-12 Computer Science Framework, 2016). The differences and relationship with computer science are described below.

- ***Computer literacy*** refers to the general use of computers and programs, such as productivity software. Examples include performing an Internet search and creating a digital presentation.
- ***Educational technology*** applies computer literacy to school subjects. For example, students in an English class can use a web-based application to collaboratively create, edit, and store an essay online.
- ***Digital citizenship*** refers to the appropriate and responsible use of technology, such as choosing an appropriate password and keeping it secure.
- ***Information technology*** often overlaps with computer science but is mainly focused on industrial applications of computer science, such as installing software rather than creating it. Information technology professionals often have a background in computer science.

Focus On Equity

Computer Science education has a history of significant access challenges, especially for early elementary students, students with disabilities, and women & minority students [csta-https://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/CSTAPolicyBrochure.pdf]. These Computer Science standards are intended to close the access gap for underserved populations and provide a foundation in computer science to *all* Arizona students.

All students and teachers have the opportunity to engage in rigorous, robust computer science standards. Each standard provides additional guidance and examples for implementation. Many standards include guidance and examples for use without computing devices, allowing for flexible implementation in lesson design and delivery.

The Arizona Computer Science Standards provide flexibility to allow students to demonstrate proficiency in multiple ways, thus providing a rigorous opportunity for engagement in computer science.

Acknowledgements

Numerous existing sets of standards and standards-related documents have been used in developing the Arizona Computer Science Standards. These include:

- The (Interim) CSTA K-12 Computer Science Standards, revised 2016 http://www.csteachers.org/?page=CSTA_Standards
- The K-12 Computer Science Framework https://k12cs.org/
- Approved or draft standards from the following states:
    - Nevada: http://www.doe.nv.gov/uploadedFiles/nde.doe.nv.gov/content/Standards_Instructional_Support/Nevada_Academic_Standards/Comp_Tech_Standards/DRAFTNevadaK-12ComputerScienceStandards.pdf
    - Wisconsin: https://dpi.wi.gov/sites/default/files/imce/computer-science/ComputerScienceStandardsFINALADOPTED.pdf

Computer Science Essential Concepts and Subconcepts

The Arizona Computer Science Standards for grades kindergarten through twelve are organized into five Essential Concepts:

- **Computing Systems:** This involves the interaction that people have with a wide variety of computing devices that collect, store, analyze, and act upon information in ways that can affect human capabilities both positively and negatively. The physical components (hardware) and instructions (software) that make up a computing system communicate and process information in digital form. An understanding of hardware and software is useful when troubleshooting a computing system that does not work as intended.

- **Networks and the Internet (with Cybersecurity):** This involves the networks that connect computing systems. Computing devices do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing. Networks and communication systems provide greater connectivity in the computing world by providing fast, secure communication and facilitating innovation. Networking and the Internet must also consider Cybersecurity. Cybersecurity, also known as information technology security, involves the protection of computers, networks, programs, and data from unauthorized or unintentional access, manipulation, or destruction. Many organizations, such as government, military, corporations, financial institutions, hospitals, and others collect, process, and store significant amounts of data on computing devices. That data is transmitted across multiple networks to other computing devices. The confidential nature of government, financial, and other types of data requires continual monitoring and protection for the sake of continued operation of vital systems and national security.

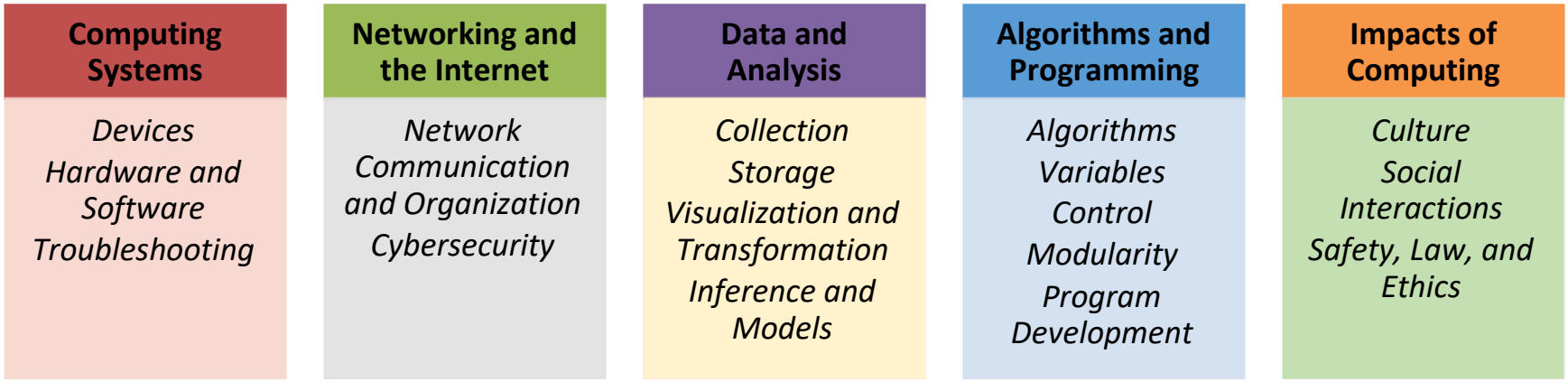
- **Data and Analysis:** This involves the data that exist and the computing systems that exist to process that data. The amount of digital data generated in the world is rapidly expanding, so the need to process data effectively is increasingly important. Data is collected and stored so that it can be analyzed to better understand the world and make more accurate predictions.

- **Algorithms and Programming:** Involves the use of algorithms. An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices. Algorithms and programming control all computing systems, empowering people to communicate with the world in new ways and solve compelling problems. The development process to create meaningful and efficient programs involves choosing which information to use and how to process and store it, breaking apart large problems into smaller ones, recombining existing solutions, and analyzing different solutions.

- **Impacts of Computing:** This involves the effect that computing has on daily life. Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and responsible person should understand the social implications of the digital world, including equity and access to computing.

Concepts are categories that represent major content areas in the field of computer science. They represent specific areas of disciplinary importance rather than abstract, general ideas. Each essential concept is supported by various subconcepts that represent specific ideas within each concept. Figure 1 provides a visual representation of the Essential Concepts and the supporting subconcepts.

Figure 1: Computer science essential concepts and subconcepts



Computer Science Practices for Students

The content of the Arizona Computer Science Standards is intended to support the following seven practices for students. The practices describe the behaviors and ways of thinking that computationally literate students use to fully engage in a data-rich and interconnected world.

- **Fostering an Inclusive Computing Culture:** Students will develop skills for building an inclusive and diverse computing culture, which requires strategies for incorporating perspectives from people of different genders, ethnicities, and abilities. Incorporating these perspectives involves understanding the personal, ethical, social, economic, and cultural contexts in which people operate. Considering the needs of diverse users during the design process is essential to producing inclusive computational products.

- **Collaborating Around Computing:** Students will develop skills for collaborating around computing. Collaborative computing is the process of performing a computational task by working in pairs and on teams. Collaborative computing involves asking for the contributions and feedback of others, effective collaboration can lead to better outcomes than working independently. Collaboration requires individuals to navigate and incorporate diverse perspectives, conflicting ideas, disparate skills, and distinct personalities. Students should use collaborative tools to effectively work together and to create complex artifacts.

- **Recognizing and Defining Computational Problems:** Students will develop skills for recognizing and defining computational problems. The ability to recognize appropriate and worthwhile opportunities to apply computation is a skill that develops over time and is central to computing. Solving a problem with a computational approach requires defining the problem, breaking it down into parts, and evaluating each part to determine whether a computational solution is appropriate.

- **Developing and Using Abstractions**: Students will develop skills for developing and using abstractions. Identifying patterns and extracting common features from specific examples to create generalizations form abstractions. Using generalized solutions and parts of solutions designed for broad reuse simplifies the development process by managing complexity.

- **Creating Computational Artifacts:** Students will develop skills for creating computational artifacts. The process of developing computational artifacts embraces both creative expression and the exploration of ideas to create prototypes and solve computational problems. Students create artifacts that are personally relevant or beneficial to their community and beyond. Computational artifacts can be created by combining and modifying existing artifacts or by developing new artifacts. Examples of computational artifacts include programs, simulations, visualizations, digital animations, robotic systems, and apps.

- **Testing and Refining Computational Artifacts:** Students will develop skills for testing and refining computational artifacts. Testing and refinement is the deliberate and iterative process of improving a computational artifact. This process includes debugging (identifying and fixing errors) and comparing actual outcomes to intended outcomes. Students also respond to the changing needs and expectations of end users and improve the performance, reliability, usability, and accessibility of artifacts.

- **Communicating About Computing**: Students will develop skills for communicating about computing. Communication involves personal expression and exchanging ideas with others. In computer science, students communicate with diverse audiences about the use and effects of computation and the appropriateness of computational choices. Students write clear comments, document their work, and communicate their ideas through multiple forms of media. Clear communication includes using precise language and carefully considering possible audiences.

See **Appendix B-Computer Science Practices,** for a complete, numbered listing of the sub-practices under each practice.

Regarding the previously listed practices, computational thinking is integrated throughout each one. Computational thinking is an approach to solving problems in a way that can be implemented with a computer. It involves the use of concepts, such as *abstraction, recursion, and iteration*, to process and analyze data, and to create real and virtual artifacts (Computer Science Teachers Association & Association for Computing Machinery, 2017). Computational thinking practices such as abstraction, modeling, and decomposition connect with computer science concepts such as algorithms, automation, and data visualization. Beginning with the elementary school grades and continuing through grade 12, students should develop a foundation of computer science knowledge and learn new approaches to problem solving that captures the power of computational thinking to become both users and creators of computing technology. Figure 2 is a visual representation of the essential practices along with computational thinking.

Figure 2: Essential Practices including computational thinking

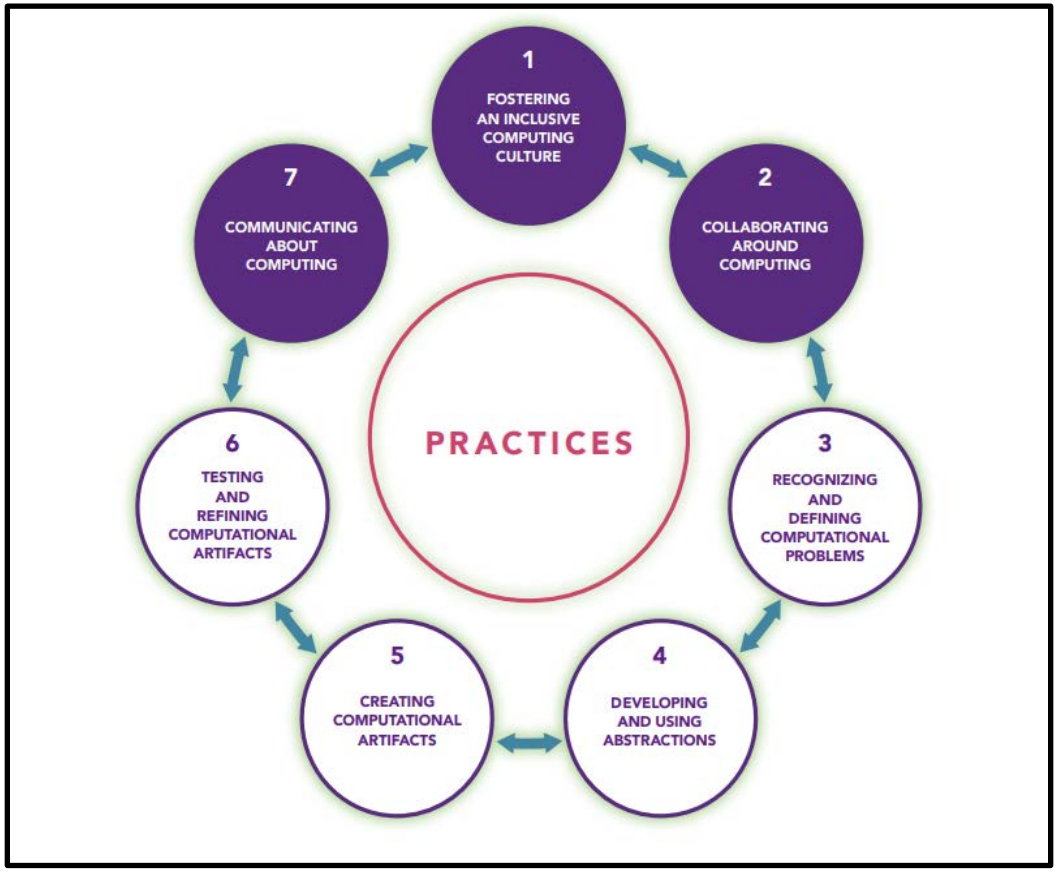
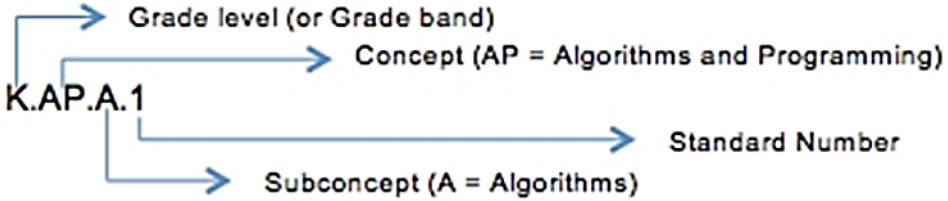


Figure 2: Practices- K-12 Computer Science Framework. (2016)

How to Read the Arizona Computer Science Standards

The Arizona Computer Science Standards are divided into Grades K, 1, 2, 3, 4, 5, 6, 7, 8, and 9-12. The standards are divided by the five main concepts. Those main concepts include computing systems, networks and the internet, data and analysis, algorithms and programming, and impacts of computing. Within each main concept there may be two to five sub concepts represented, such as algorithms, program development, variables, troubleshooting, or cybersecurity. Each standard will list the grade level, the concept, the subconcept, and the standard number. Figure 3 provides an example of the coding for a standard:

Figure 3: Standard Coding Scheme for Standards



Each standard appears like the example in Figure 4. This example shows two different subconcepts under the concept of Algorithms and Programming at the Kindergarten level.

Figure 4: Example of Complete Individual Standard

Subconcept: Algorithms	
K.AP.A.1	**With teacher assistance, model daily processes by following algorithms (sets of step-by-step instructions) to complete tasks.** *Routines, such as morning meeting, clean-up time, and dismissal, are examples of algorithms that are common in many early elementary classrooms. Just as people use algorithms to complete daily routines, they can program computers to use algorithms to complete different tasks. Algorithms are commonly implemented using a precise language that computers can interpret. For example, students begin to recognize daily step-by-step processes, such as brushing teeth or following a morning procedure, as "algorithms" that lead to an end result.* *Practice(s):* Developing and Using Abstractions: 4.4
Subconcept: Variables	
K.AP.V.1	**With teacher assistance, model the way programs store and manipulate data by using numbers or other symbols to represent information.** *Information in the real world can be represented in computer programs. Students could use thumbs up/down as representations of yes/no, use arrows when writing algorithms to represent direction, or encode and decode words using numbers, pictographs, or other symbols to represent letters or words.* Practice(s): Developing and Using Abstractions: 4.4

Kindergarten

Since the standards drive the pedagogical component of teaching and serve as a guide to how students demonstrate understanding of the content, they must be incorporated as an integral part of the overall student expectations when assessing content understanding. By the end of Kindergarten, the computer science literate student will begin the practice of utilizing devices to perform basic computer operations, such as turning a computer on and off, and communicate basic hardware and software problems. Kindergarten students will be able to explain the importance of password protection and discuss how computer networks can connect people globally. With teacher assistance, kindergarten students will begin to develop an understanding of how to model and develop algorithms and programs, and collect data to make predictions. Students will be introduced to the impacts of computing, including how people lived and worked before and after the implementation of new technology, how to work responsibly online, and the importance of keeping login information private.

Concept: Computing Systems (CS)

Subconcept: Devices (D)	
K.CS.D.1	**With teacher guidance, select and operate an appropriate device to perform a task.** *People use computing devices to perform a variety of tasks accurately and quickly. With teacher guidance, students should be able to select the appropriate device to use for tasks they are required to complete. For example, if students are asked to draw a picture, they should be able to open and use a drawing app/program to complete this task, or if they are asked to create a presentation, they should be able to open and use presentation software.* Practice(s): Fostering an inclusive Computing Culture: 1.2
Subconcept: Hardware and Software (HS)	
K.CS.HS.1	**Use appropriate terminology in identifying and describing the function of common physical components of computing systems.** *A computing system is composed of hardware and software. Hardware consists of physical components. Students should be able to identify and describe the function of external hardware, such as desktop computers, laptop computers, tablet devices, monitors, keyboards, mice, and printers.* Practice(s): Communicating about Computing: 7.2
Subconcept: Troubleshooting (T)	
K.CS.T.1	**Discuss basic hardware and software problems.** *Problems with computing systems have different causes. Students should be able to communicate a hardware or software problem (e.g., when an app or program is not working as expected, a device will not turn on, the sound does not work, etc.).* Practice(s): Testing and Refining Computational Artifacts, Communicating About Computing: 6.2, 7.3

Concept: Networks and the Internet (NI)

Subconcept: Cybersecurity (C)	
K.NI.C.1	**Explain that a password helps protect the privacy of information.** *Connecting devices to a network or the Internet provides great benefit, care must be taken to use authentication measures, such as strong passwords, to protect devices and information from unauthorized access. This is an essential first step in learning about cybersecurity. Usernames and passwords, such as those on computing devices or Wi-Fi networks, provide a way of authenticating a user's identity. For example, students should enter a password independently and commit to keeping their password private.* *Practice(s):* Communicating About Computing: 7.2
Subconcept: Network, Communication, and Organization (NCO)	
K.NI. NCO.1	**With teacher guidance, students define computer networks and how they can be used to connect people to other people, places, information, and ideas.** *Small, wireless devices, such as cell phones, communicate with one another through a series of intermediary connection points, such as cellular towers. This coordination among many computing devices allows a person to voice call a friend or video chat with a family member. For example, kindergarten students understand that they are part of non-computing networks such as family, class, school, etc. Kindergarten students should be able to explain that devices are connected, though details about connection points are not expected at this level.* *Practice(s): Communicating About Computing: 7.3*

Concept: Data and Analysis (DA)

Subconcept: Collection, Visualization and Transformation (CVT)	
K.DA. CVT.1	**With teacher guidance, collect and transform data using digital devices; Display data for communication in various visual formats.** *The collection and use of data about the world around them is a routine part of life and influences how people live. Many everyday objects, such as cell phones, digital toys, and cars, can contain tools (such as sensors) and computers to collect and display data from their surroundings. Students could collect data on the weather, such as sunny days versus rainy days, the temperature at the beginning of the school day and end of the school day, or the inches of rain over the course of a storm. Students could count the number of pieces of each color of candy in a bag of candy, such as Skittles or M&Ms. Students could create surveys of things that interest them, such as favorite foods, pets, or TV shows, and collect answers to their surveys from their peers and others. The data collected could then be organized into two or more visualizations, such as a bar graph, pie chart, or pictograph.* *Practice(s):* Communicating About Computing, Developing and Using Abstractions: 7.3, 4.4

Subconcept: Storage (S)	
K.DA.S.1	**Recognize that data can be collected and stored on different computing devices over time and retrieved later.** *All information stored and processed by a computing device is referred to as data. Data can be images, text documents, audio files, software programs or apps, video files, etc. It can be retrieved, copied, and stored in multiple places. As students use software to complete tasks on a computing device, they will be manipulating data. For example, students should be able to create and save a document.* *Practice(s):* Developing and Using Abstractions: 4.3
Subconcept: Inference and Models (IM)	
K.DA.IM.1	**Discuss patterns in data to make inferences or predictions.** *Data can be used to make inferences or predictions about the world. Students could analyze a Graph and pie chart of the colors in a bag of candy, identify which colors are most and least represented, and then make a prediction as to which colors will have most and least in a new bag of candy. For example, students preview a weather graph for one week in their city and make predictions about the weather for the following week.* *Practice(s):* Developing and Using Abstractions: 4.1

Concept: Algorithms and Programming (AP)

Subconcept: Algorithms (A)	
K.AP.A.1	**With teacher assistance, model daily processes by following algorithms (sets of step-by-step instructions) to complete tasks.** *Routines, such as morning meeting, clean-up time, and dismissal, are examples of algorithms that are common in many early elementary classrooms. Just as people use algorithms to complete daily routines, they can program computers to use algorithms to complete different tasks. Algorithms are commonly implemented using a precise language that computers can interpret. For example, students begin to recognize daily step-by-step processes, such as brushing teeth or following a morning procedure, as "algorithms" that lead to an end result.* *Practice(s):* Developing and Using Abstractions: 4.4
Subconcept: Variables (V)	
K.AP.V.1	**With teacher assistance, model the way programs store and manipulate data by using numbers or other symbols to represent information.** *Information in the real world can be represented in computer programs. Students could use thumbs up/down as representations of yes/no, use arrows when writing algorithms to represent direction, or encode and decode words using numbers, pictographs, or other symbols to represent letters or words.* Practice(s): Developing and Using Abstractions: 4.4

Subconcept: Control (C)	
K.AP.C.1	**With teacher assistance, identify programs with sequences and simple loops, to express ideas or address a problem.** *Programming is used as a tool to create products that reflect a wide range of interests. Control structures specify the order in which instructions are executed within a program. Computers follow instructions literally. Sequences are the order of instructions in a program. For example, sequences of instructions include steps for drawing a shape or moving a character across the screen. If the commands to program a robot are not in the correct order, the robot will not complete the task desired. Loops allow for the repetition of a sequence of code multiple times. For example, in a program to show the life cycle of a butterfly, a loop could be combined with move commands to allow continual but controlled movement of the character. For example, kindergarten students should be able to recognize loops and sequences in songs, rhymes, and games, such as the song B-I-N-G-O.* Practice(s): Creating Computational Artifacts: 5.1
Subconcept: Modularity (M)	
K.AP.M.1	**With teacher assistance, solve a problem by breaking it down into smaller parts.** *Decomposition is the act of breaking down tasks into simpler tasks. For example, Students could break down the steps needed to make a peanut butter and jelly sandwich, to brush their teeth, to draw a shape, to move a character across the screen, or to solve a level of a coding app.* Practice(s): Recognizing and Defining Computational Problems: 3.1
Subconcept: Program Development (PD)	
K.AP.PD.1	**With teacher assistance, develop plans that describe a program's sequence of events, goals, and expected outcomes.** *Programming is used as a tool to create products that reflect a wide range of interests, such as video games, interactive art projects, and digital stories. Students could create a planning document, such as a story map, a storyboard, or a sequential graphic organizer, to illustrate what an end product will do. Students at this stage may complete the planning process with help from their teachers. For example, kindergarten students could illustrate the beginning, middle, and end of a favorite story.* Practice(s): Creating Computational Artifacts, Communicating About Computing: 5.1, 7.2
K.AP.PD.2	**With teacher assistance, identify attribution (credit) when using the ideas and creations of others while developing programs.** *Using computers comes with a level of responsibility. Students should credit artifacts that were created by others, such as pictures, music, and code. Credit could be given orally, if presenting their work to the class, or in writing or orally, if sharing work on a class blog or website. Proper attribution at this stage does not require a formal citation, such as in a bibliography or works cited document.* Practice(s): Communicating About Computing: 7.3
K.AP.PD.3	**With teacher assistance, debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.** *Algorithms or programs may not always work correctly. Students should be able to use various strategies, such as changing the sequence of the steps, following the algorithm in a step-by-step manner, or trial and error to fix problems in algorithms and programs. For example, kindergarten students should be able to identify incorrect order in a series of events and place them in the correct order, such as getting ready for school or making a peanut butter sandwich.* Practice(s): Testing and Refining Computational Artifacts: 6.2

K.AP.PD.4	**With teacher assistance, using correct terminology, describe steps taken and choices made during program development.** *At this stage, students should be able to talk or write about the goals and expected outcomes of the instructions they develop they create and the choices that they made when developing their instructions. This could be done using coding journals, discussions with a teacher, or teacher created classroom blogs. For example, kindergarten students could describe their thinking about a story map or set of instructions they develop.* *Practice(s)*: Communicating About Computing: 7.2

Concept: Impacts of Computing (IC)

Subconcept: Culture (C)	
K.IC.C.1	**Discuss how people lived and worked before and after the implementation or adoption of new computing technology.** *Computing technology has positively and negatively changed the way people live and work. In the past, if students wanted to read about a topic, they needed access to a library to find a book about it. Today, students can view information on the Internet about a topic or they can download e-books about it directly to a device.* Practice(s): Communicating About Computing: 7.2
Subconcept: Social Interactions (SI)	
K.IC.SI.1	**Work respectfully and responsibly with others online.** *Online communication facilitates positive interactions, such as sharing ideas with many people, but the public and anonymous nature of online communication also allows intimidating and inappropriate behavior in the form of cyberbullying. Teachers should facilitate a discussion in how to avoid sharing information that is inappropriate or that could personally identify them to others, and how to. work in a kind and respectful manner.* Practice(s): Collaborating Around Computing: 2.1
Subconcept: Safety, Law, and Ethics (SLE)	
K.IC. SLE.1	**Keep login information private, and log off of devices appropriately.** *Using computers comes with a level of responsibility. Students should not share login information, keep passwords private, and log off when finished* Practice(s): Communicating About Computing:7.2

First Grade

Students learn foundational concepts by integrating basic digital literacy skills with simple ideas about computational thinking. Students learn that tools help people do things better, or more easily, or do some things that could otherwise not be done at all. Through the exploration of differences between humans, computing devices, and digital tools, students begin to understand if, when, and how they should use technology. By the end of first grade, the computer science literate student will recognize user needs and preferences while utilizing devices to perform basic computer operations, hardware, software, and apply basic troubleshooting strategies. Students will explain and practice the importance of password protection and discuss how computer networks can connect people globally. With teacher guidance, students will collect, transform, and explain how different types of data can be stored and retrieved from a computing device. First grade students will develop an understanding of how to model and identify algorithms and programs using loops and step by step instructions. First grade students will discuss the impacts of computing, including how people lived and worked before and after the implementation of new technology, how to work responsibly online, the importance of keeping login information private and logging off devices appropriately.

Concept: Computing Systems (CS)

Subconcept: Devices (D)	
1.CS.D.1	**With teacher guidance, select and operate appropriate devices and software to perform a task.** *People use computing devices to perform a variety of tasks accurately and quickly. With teacher guidance students should be able to select the appropriate app/program to use for tasks they are required to complete. For example, if students are asked to draw a picture, they should be able to open and use a drawing app/program to complete this task.* Practice(s): Fostering an inclusive Computing Culture, Communicating About Computing: 1.2, 7.3
Subconcept: Hardware and Software (HS)	
1.CS.HS.1	**Use appropriate terminology in identifying and describing the function of common physical components of computing systems.** *A computing system is composed of hardware and software. Hardware consists of physical components. Students should be able to identify and describe the function of external hardware, such as desktop computers, laptop computers, tablet devices, monitors, keyboards, mice, and printers. Students should be able to identify software such as: web browsers, games, etc.* Practice(s): Communicating about Computing: 7.2

Subconcept: Troubleshooting (T)	
1.CS.T.1	**Identify basic hardware and software problems using accurate terminology.** *Problems with computing systems have different causes. Students should be able to communicate a hardware or software problem with accurate terminology (e.g., when an app or program is not working as expected, a device will not turn on, the sound does not work, etc.).* Practice(s): Testing and Refining Computational Artifacts, Communicating About Computing: 6.3, 7.2
1.CS.T.2	**With teacher guidance, begin to use basic troubleshooting strategies.** *Students would be able to use simple troubleshooting strategies. For example, turning a device off and on to reboot it, closing and reopening an app, turning on speakers, or plugging in headphones, and then adjusting volume.* Practice(s): Testing and Refining Computational Artifacts: 6.2

Concept: Networks and the Internet (NI)

Subconcept: Cybersecurity (C)	
1.NI.C.1	**Explain what passwords are and why we use them to protect personal information (e.g., name, location, phone number, home address) and keep it private.** *Connecting devices to a network or the Internet provides great benefit, care must be taken to use authentication measures, such as strong passwords, to protect devices and information from unauthorized access. This is an essential first step in learning about cybersecurity. For example, first grade students should be able to accurately enter a password to log on to a program and understand the importance of keeping passwords private in order to protect their personal information.* *Practice(s):* Communicating About Computing: 7.2
Subconcept: Network, Communication, and Organization (NCO)	
1.NI. NCO.1	**With teacher guidance, students discuss how computer networks can be used to connect people to other people, places, information, and ideas.** *Small, wireless devices, such as cell phones, communicate with one another through a series of intermediary connection points, such as cellular towers. This coordination among many computing devices allows a person to voice call a friend or video chat with a family member. Details about the connection points are not expected at this level. For example, students will participate in a class discussion about how different networks connect people, places, things and information, such as a phone call to grandma in another state, using Facetime or Skype to connect with a content area expert, connecting devices via Bluetooth, or accessing an online game through Wi-Fi.* *Practice(s): Communicating About Computing: 7.2*

Concept: Data and Analysis (DA)

Subconcept: Collection, Visualization and Transformation (CVT)	
1.DA.CVT.1	**With teacher guidance, collect and transform data using digital devices; Display data for communication in various visual formats.** *The collection and use of data about the world around them is a routine part of life and influences how people live. Many everyday objects, such as cell phones, digital toys, and cars, can contain tools (such as sensors) and computers to collect and display data from their surroundings. Students could collect data on the weather, such as sunny days versus rainy days, the temperature at the beginning of the school day and end of the school day, or the inches of rain over the course of a storm. Students could count the number of pieces of each color of candy in a bag of candy, such as Skittles or M&Ms. Students could create surveys of things that interest them, such as favorite foods, pets, or TV shows, and collect answers to their surveys from their peers and others. The data collected could then be organized into two or more visualizations, such as a bar graph, pie chart, or pictograph.* *Practice(s):* Communicating About Computing, Developing and Using Abstractions: 7.1, 4.2
Subconcept: Storage (S)	
1.DA.S.1	**Explain that a variety of data (e.g., music, video, images, and text) can be stored in and retrieved from a computing device.** All information stored and processed by a computing device is referred to as data. Data can be images, text documents, audio files, software programs or apps, video files, etc. It can be retrieved, copied, and stored in multiple places. As students use software to complete tasks on a computing device, they will be manipulating data. For example, first graders should be able to retrieve files that they previously created and saved, such as, locating and opening a word processing program they saved the previous day. *Practice(s):* Developing and Using Abstractions: 4.3
Subconcept: Inference and Models (IM)	
1.DA.IM.1	**Identify patterns in data to make inferences or predictions.** *Data can be used to make inferences or predictions about the world. Students could analyze a Graph and pie chart of the colors in a bag of candy, identify the patterns for which colors are most and least represented, and then make a prediction as to which colors will have most and least in a new bag of candy.* *Practice(s):* Developing and Using Abstractions: 4.4

Concept: Algorithms and Programming (AP)

Subconcept: Algorithms (A)	
1.AP.A.1	**Model daily processes by following algorithms (sets of step-by-step instructions) to complete tasks.** *Routines, such as morning meeting, clean-up time, and dismissal, are examples of algorithms that are common in many early elementary classrooms. For example, students begin to understand and model daily step-by-step processes, such as brushing teeth, implementing a morning procedure, or following a simple recipe as "algorithms" that lead to an end result.* *Practice(s):* Developing and Using Abstractions: 4.4
Subconcept: Variables (V)	
1.AP.V.1	**Model the way programs store and manipulate data by using numbers or other symbols to represent information.** *Information in the real world can be represented in computer programs. Students could use thumbs up/down as representations of yes/no, use arrows when writing algorithms to represent direction, or encode and decode words using numbers, pictographs, or other symbols to represent letters or words.* *Practice(s):* Developing and Using Abstractions: 4.3
Subconcept: Control (C)	
1.AP.C.1	**Identify programs with sequences and simple loops, to express ideas or address a problem.** *Programming is used as a tool to create products that reflect a wide range of interests. Control structures specify the order in which instructions are executed within a program. Computers follow instructions literally. Sequences are the order of instructions in a program. For example, sequences of instructions include steps for drawing a shape or moving a character across the screen. If the commands to program a robot are not in the correct order, the robot will not complete the task desired. Loops allow for the repetition of a sequence of code multiple times. For example, in a program to show the life cycle of a butterfly, a loop could be combined with move commands to allow continual but controlled movement of the character. For example, first grade students independently identify loops and sequences in songs, rhymes, and games, such as the song B-I-N-G-O or the game Red Light/Green Light.* *Practice(s):* Creating Computational Artifacts: 5.1
Subconcept: Modularity (M)	
1.AP.M.1	**Solve a problem by breaking it down into smaller parts.** *Decomposition is the act of breaking down tasks into simpler tasks. For example, Students could break down the steps needed to make a peanut butter and jelly sandwich, to brush their teeth, to draw a shape, to move a character across the screen, or to solve a level of a coding app.* *Practice(s):* Recognizing and Defining Computational Problems: 3.1

Subconcept: Program Development (PD)	
1.AP.PD.1	**With teacher assistance identify plans that describe a program's sequence of events, goals, and expected outcomes.** *Programming is used as a tool to create products that reflect a wide range of interests, such as video games, interactive art projects, and digital stories. Students could create a planning document, such as a story map, a storyboard, or a sequential graphic organizer, to illustrate what their end product will do. Students at this stage may complete the planning process with help from their teachers. For example, students create a comic strip with at least 3 panels showing the sequence of a story.* *Practice(s):* Creating Computational Artifacts, Communicating About Computing: 5.3, 7.1
1.AP.PD.2	**With teacher assistance, give attribution (credit) when using the ideas and creations of others while developing programs.** *Using computers comes with a level of responsibility. Students should credit artifacts that were created by others, such as pictures, music, and code. Credit could be given orally, if presenting their work to the class, or in writing or orally, if sharing work on a class blog or website. Proper attribution at this stage does not require a formal citation, such as in a bibliography or works cited document.* *Practice(s):* Communicating About Computing: 7.3
1.AP.PD.3	**With teacher assistance, debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.** *Algorithms or programs may not always work correctly. Students should be able to use various strategies, such as changing the sequence of the steps, following the algorithm in a step-by-step manner, or trial and error to fix problems in algorithms and programs. For example, first graders should be able to identify and fix incorrect order in a series of events, placing them in the correct order, such as washing dishes at home. When the steps repeat, a loop is created.* *Practice(s):* Testing and Refining Computational Artifacts: 6.3
1.AP.PD.4	**Using correct terminology, describe steps taken and choices made during program development.** *At this stage, students should be able to talk or write about the goals and expected outcomes of the instructions they develop and the choices that they made when developing their instructions. This could be done using coding journals, discussions with a teacher, class presentations, or classroom blogs. For example, first grade students do a class presentation sharing the process, choices they made, and outcomes for a fictitious product they developed.* *Practice(s):* Communicating About Computing: 7.2

Concept: Impacts of Computing (IC)

Subconcept: Culture (C)	
1.IC.C.1	**Discuss how people live and work before and after the implementation or adoption of new computing technology.** *Computing technology has positively and negatively changed the way people live and work. In the past, if students wanted to read about a topic, they needed access to a library to find a book about it. Today, students can view and read information on the Internet about a topic or they can download e-books about it directly to a device. Such information may be available in more than one language and could be read to a student, allowing for great accessibility.* *Practice(s):* Communicating About Computing: 7.1

Subconcept: Social Interactions (SI)	
1.IC.SI.1	**Work respectfully and responsibly with others online.** *Online communication facilitates positive interactions, such as sharing ideas with many people, but the public and anonymous nature of online communication also allows intimidating and inappropriate behavior in the form of cyberbullying. Students could share their work on blogs or in other collaborative spaces online, taking care to avoid sharing information that is inappropriate or that could personally identify them to others. Students could provide feedback to others on their work in a kind and respectful manner. They should tell an adult if others are sharing things they should not share or are treating others in an unkind or disrespectful manner on online. Privacy should be considered when posting information online: such information can persist for a long time and be accessed by others, even unintended viewers.* *Practice(s):* Collaborating Around Computing: 2.1
Subconcept: Safety, Law, and Ethics (SLE)	
1.IC. SLE.1	**Keep login information private, and log off devices appropriately.** *Using computers comes with a level of responsibility, such as not sharing login information, keeping passwords private, and logging off when finished. Rules guiding personal interactions in the world, apply to online environments as well. For example, students routinely practice logging in and logging out of online resources to protect their personal information. Students should also commit to interacting with only those they know in person in online environments.* *Practice(s):* Communicating About Computing: 7.2

Second Grade

Students expand basic knowledge of digital literacy and computer science skills. By the end of second grade, students will be able to select the appropriate device and software to perform specific tasks. Students will understand that computing systems use both hardware and software to process information and will be able to use basic troubleshooting strategies with teacher guidance. Students will develop a deeper understanding of the importance and use of strong passwords to protect private information and discuss how computer networks can connect people globally. Second grade students will independently collect, transform, and display data using digital devices. Students can store, copy, search, retrieve, modify, and delete information using a computing device. Second grade students will create and follow algorithms and programs using loops and variables to solve a problem. Students will decompose steps into simpler tasks and give credit to the ideas and creations of others. Second grade students will discuss the impacts of computing by comparing how people lived and worked before and after the implementation of new technology, how to work responsibly online, the importance of keeping login information private, and logging off devices appropriately.

Concept: Computing Systems (CS)

Subconcept: Devices (D)	
2.CS.D.1	**Recognize that users have different needs and preferences for technology they used by selecting and operating appropriate devices.** *People use computing devices to perform a variety of tasks accurately and quickly. Students should be able to select the appropriate app/program to use for tasks they are required to complete. For example, if students are asked to draw a picture, they should be able to open and use a drawing app/program to complete this task, or if they are asked to create a presentation, they should be able to open and use presentation software. In addition, with teacher guidance, students should identify and discuss preferences for software with the same primary functionality.* *Practice(s):* Fostering an inclusive Computing Culture, Communicating About Computing: 1.1, 7.3
Subconcept: Hardware and Software (HS)	
2.CS.HS.1	**Understand how computing systems use both hardware (device) and software (program/app) to process information.** *A computing system is composed of hardware and software. Hardware consists of physical components. Students should be able to identify and describe the function of external hardware, such as desktop computers, laptop computers, tablet devices, monitors, keyboards, mice, and printers.* *Practice(s):* Communicating about Computing : 7.2

Subconcept: Troubleshooting (T)	
2.CS.T.1	**Explain basic hardware (device) and software (program/app) problems using accurate terminology.** *Problems with computing systems have different causes. Students at this level do not need to understand those causes, but they should be able to communicate a problem with accurate terminology (e.g., when an app or program is not working as expected, the device is frozen, the link doesn't work, the internet is not connecting).* *Practice(s):* Communicating About Computing: 7.2
2.CS.T.2	**With teacher guidance, use basic troubleshooting strategies.** *Students should be able to use simple troubleshooting strategies, including turning a device off and on to reboot it, closing and reopening an app, turning on speakers, or plugging in headphones and adjusting volume.* *Practice(s):* Testing and Refining Computational Artifacts: 6.2

Concept: Networks and the Internet (NI)

Subconcept: Cybersecurity (C)	
2.NI.C.1	**Explain what passwords are and why we use them, and use strong passwords to protect devices and information from unauthorized access.** *Connecting devices to a network or the Internet provides great benefit, care must be taken to use authentication measures, such as strong passwords, to protect devices and information from unauthorized access. This is an essential first step in learning about cybersecurity. They should appropriately use and protect the passwords they are required to use. Usernames and passwords, such as those on computing devices or Wi-Fi networks, provide a way of authenticating a user's identity. For example, students learn to not share passwords and not use anyone else's password.* *Practice(s):* Communicating About Computing: 7.2
Subconcept: Network, Communication, and Organization (NCO)	
2.NI. NCO.1	**Students can discuss how computer networks can be used to connect people to other people, places, information, and ideas.** *Small, wireless devices, such as cell phones, communicate with one another through a series of intermediary connection points, such as cellular towers. This coordination among many computing devices allows a person to voice call a friend or video chat with a family member. Details about the connection points are not expected at this level. For example, students will participate in a class discussion about how different networks connect people, places, things and information, such as a phone call to grandma in another state, using conferencing software to connect with a content area expert, or accessing an online game via Wi-Fi.* *Practice(s): Communicating About Computing: 7.2*

Concept: Data and Analysis (DA)

Subconcept: Collection, Visualization and Transformation (CVT)	
2.DA.CVT.1	**Collect and transform data using digital devices; Display data for communication in various visual formats.** *The collection and use of data about the world around them is a routine part of life and influences how people live. Many everyday objects, such as cell phones, digital toys, and cars, can contain tools (such as sensors) and computers to collect and display data from their surroundings. Students could collect data on the weather, such as sunny days versus rainy days, the temperature at the beginning of the school day and end of the school day, or the inches of rain over the course of a storm. Students could count the number of pieces of each color of candy in a bag of candy, such as Skittles or M&Ms. Students could create surveys of things that interest them, such as favorite foods, pets, or TV shows, and collect answers to their surveys from their peers and others. The data collected could then be organized into two or more visualizations, such as a bar graph, pie chart, or pictograph.* *Practice(s):* Communicating About Computing, Developing and Using Abstractions: 7.3, 4.2
Subconcept: Storage (S)	
2.DA.S.1	**Store, copy, search, retrieve, modify, and delete information using a computing device and define the information stored as data.** All information stored and processed by a computing device is referred to as data. Data can be images, text documents, audio files, software programs or apps, video files, etc. It can be retrieved, copied, and stored in multiple places. As students use software to complete tasks on a computing device, they will be manipulating data. For example, students will learn to save files in specific locations, such as a folder, and retrieve those files for use later. *Practice(s):* Developing and Using Abstractions: 4.1
Subconcept: Inference and Models (IM)	
2.DA.IM.1	**Describe patterns in data to make inferences or predictions.** *Data can be used to make inferences or predictions about the world. Students could analyze a Graph and pie chart of the colors in a bag of candy identify the patterns for which colors are most and least represented, and then make a prediction as to which colors will have most and least in a new bag of candy.* *Practice(s):* Developing and Using Abstractions: 4.3

Concept: Algorithms and Programming (AP)

Subconcept: Algorithms (A)	
2.AP.A.1	**Model daily processes by creating and following algorithms (sets of step-by-step instructions to complete tasks.** *Routines, such as morning meeting, clean-up time, and dismissal, are examples of algorithms that are common in many early elementary classrooms. Just as people use algorithms to complete daily routines, they can program computers to use algorithms to complete different tasks. Algorithms are commonly implemented using a precise language that computers can interpret. For example, students begin to understand and model daily step-by-step processes, such as brushing teeth, implementing a morning procedure, or following a simple recipe as "algorithms" that lead to an end result.* *Practice(s):* Developing and Using Abstractions: 4.3
Subconcept: Variables (V)	
2.AP.V.1	**Model the way programs store and manipulate data by using numbers or other symbols to represent information.** *Information in the real world can be represented in computer programs. Students could use thumbs up/down as representations of yes/no, use arrows when writing algorithms to represent direction, or encode and decode words using numbers, pictographs, or other symbols to represent letters or words.* *Practice(s):* Developing and Using Abstractions: 4.3
Subconcept: Control (C)	
2.AP.C.1	**Develop programs with sequences and simple loops, to express ideas or address a problem.** *Programming is used as a tool to create products that reflect a wide range of interests. Control structures specify the order in which instructions are executed within a program. Computers follow instructions literally. Sequences are the order of instructions in a program. For example, sequences of instructions include steps for drawing a shape or moving a character across the screen. If the commands to program a robot are not in the correct order, the robot will not complete the task desired. Loops allow for the repetition of a sequence of code multiple times. For example, in a program to show the life cycle of a butterfly, a loop could be combined with move commands to allow continual but controlled movement of the character. For example: students independently identify loops and sequences in songs, rhymes, and games, such as the song Head, Shoulders, Knees and Toes* *Practice(s):* Creating Computational Artifacts: 5.2
Subconcept: Modularity (M)	
2.AP.M.1	**Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.** *Decomposition is the act of breaking down tasks into simpler tasks. Students could break down the steps needed to make a peanut butter and jelly sandwich, to brush their teeth, to draw a shape, to move a character across the screen, or to solve a level of a coding app.* *Practice(s):* Recognizing and Defining Computational Problems: 3.2

Subconcept: Program Development (PD)	
2.AP.PD.1	**Develop plans that describe a program's sequence of events, goals, and expected outcomes.** *Programming is used as a tool to create products that reflect a wide range of interests, such as video games, interactive art projects, and digital stories. Students could create a planning document, such as a story map, a storyboard, or a sequential graphic organizer, to illustrate what an end product will do. Students at this stage may complete the planning process with help from their teachers. For example, students create a graphic organizer modeling the life cycle of a plant.* *Practice(s):* Creating Computational Artifacts, Communicating About Computing: 5.2, 7.2
2.AP.PD.2	**Give attribution (credit) when using the ideas and creations of others while developing programs.** *Using computers comes with a level of responsibility. Students should credit artifacts that were created by others, such as pictures, music, and code. Proper attribution at this stage does not require a formal citation, such as in a bibliography or works cited document. For example, students can give attribution in written form, at a minimum, by listing a website where they got the information, picture, or music.* *Practice(s):* Communicating About Computing: 7.3
2.AP.PD.3	**Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.** *Algorithms or programs may not always work correctly. Students should be able to use various strategies, such as changing the sequence of the steps, following the algorithm in a step-by-step manner, or trial and error to fix problems in algorithms and programs. For example: Students could use arrows on a grid to map out a path to a specific coordinate, such as a treasure map. If the steps were repeated, it would create a loop.* *Practice(s):* Testing and Refining Computational Artifacts: 6.2
2.AP.PD.4	**Using correct terminology, describe steps taken and choices made during the iterative process of program (procedure) development.** *At this stage, students should be able to talk or write about the goals and expected outcomes of the programs they create and the choices that they made when creating programs. This could be done using coding journals, or discussions with a teacher. Students work together to explain the steps in their procedure.* *Practice(s):* Communicating About Computing: 7.2

Concept: Impacts of Computing (IC)

Subconcept: Culture (C)	
2.IC.C.1	**Compare how people live and work before and after the implementation or adoption of new computing technology.** *Computing technology has positively and negatively changed the way people live and work. In the past, if students wanted to read about a topic, they needed access to a library to find a book about it. Today, students can view and read information on the Internet about a topic or they can download e-books about it directly to a device. Such information may be available in more than one language and could be read to a student, allowing for great accessibility.* *Practice(s):* Communicating About Computing: 7.1
Subconcept: Social Interactions (SI)	
2.IC.SI.1	**Work respectfully and responsibly with others online.** *Online communication facilitates positive interactions, such as sharing ideas with many people, but the public and anonymous nature of online communication also allows intimidating and inappropriate behavior in the form of cyberbullying. Students could share their work on blogs or in other collaborative spaces online, taking care to avoid sharing information that is inappropriate or that could personally identify them to others. Students could provide feedback to others on their work in a kind and respectful manner. They should tell an adult if others are sharing things they should not share or are treating others in an unkind or disrespectful manner on online. Privacy should be considered when posting information online: such information can persist for a long time and be accessed by others, even unintended viewers.* *Practice(s):* Collaborating Around Computing: 2.1
Subconcept: Safety, Law, and Ethics (SLE)	
2.IC. SLE.1	**Keep login information private, and log off of devices appropriately.** *Using computers comes with a level of responsibility, such as not sharing login information, keeping passwords private, and logging off when finished. Rules guiding personal interactions in the world apply to online environments as well. For example, students routinely practice logging in and logging out of online resources to protect their personal information. Students should also commit to interacting with only those they know in person in online environments.* *Practice(s):* Communicating About Computing: 7.2

Appendix A-Computer Science Glossary

The following glossary includes definitions of commonly-used computer science terms and was borrowed (with permission) from the K–12 Computer Science Framework. This section is intended to increase teacher understanding and use of computer science terminology.

Abstraction: Pulling out specific difference to make one solution work for multiple problems.

- (Process): 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. In elementary classrooms, abstraction is hiding unnecessary details to make it easier to think about a problem.
- (Product): A new representation of a thing, a system, or a problem that helpfully reframes a problem by hiding details irrelevant to the question at hand.

Algorithm: A step-by-step process to complete a task. A list of steps to finish a task. A set of instructions that can be performed with or without a computer.

For example, the collection of steps to make a peanut butter and jelly sandwich is an algorithm.

App: A type of application software, often 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 computer science-specific definition.

ASCII: (American Standard Code for Information Interchange) is the most common format for text files in computers and on the Internet. In an ASCII file, each alphabetic, numeric, or special character is represented with a 7-bit binary number (a string of seven 0s or 1s). 128 possible characters are defined.

Automation: The process of linking disparate systems and software in such a way that they become self-acting or self-regulating.

Backup: The process of making copies of data or data files to use in the event the original data or data files are lost or destroyed.

Binary: A system of notation representing data using two symbols (usually 1 and 0).

Block-based programming language: Any programming language that lets users create programs by manipulating "blocks" or graphical programming elements, rather than writing code using text. (Sometimes called visual coding, drag and drop programming, or graphical programming blocks)

Bug: An error in a software program. It may cause a program to unexpectedly quit or behave in an unintended manner. The process of removing errors (bugs) is called debugging.

Cloud: Remote servers that store data and are accessed from the Internet.

Code: Any set of instructions expressed in a programming language. One or more commands or algorithm(s) designed to be carried out by a computer. See also: program

Command: An instruction for the computer. Many commands put together make up algorithms and computer programs.

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 models: Used to make predictions about processes or phenomenon based on selected data and features that can be represented by organizational software.

Computational thinking: Mental processes and strategies that include: decomposition (breaking larger problems into smaller, more manageable problems), pattern matching (finding repeating patterns), abstraction (identifying specific changes that would make one solution work for multiple problems), and algorithms (a step-by-step set of instructions that can be acted upon by a computer).

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

Conditionals: Statements that only run under certain conditions or situations.

Data: Information. Often, quantities, characters, or symbols that are the inputs and outputs of computer programs.

Debugging: Finding and fixing errors in programs.

Decompose: Break a problem down into smaller pieces.

Decryption: The process of taking encoded or encrypted text or other data and converting it back into text that you or the computer can read and understand.

Digital divide: the gulf between those who have ready access to computers and the Internet, and those who do not.

Encryption: The process of encoding messages or information in such a way that only authorized parties can read it.

Event: An action that causes something to happen

Execution: The process of executing an instruction or instruction set.

For loop: A loop with a predetermined beginning, end, and increment (step interval)

Function: A type of procedure or routine. Some programming languages make a distinction between a function, which returns a value, and a procedure, which performs some operation, but does not return a value. Note: This definition differs from that used in math. A piece of code that you can easily call over and over again. Functions are sometimes called 'procedures.'

GPS: Abbreviation for "Global Positioning System." GPS is a satellite navigation system used to determine the ground position of an object.

Hacking: Appropriately applying ingenuity. Cleverly solving a programming problem. Using a computer to gain unauthorized access to data within a system.

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

Hierarchy: An organizational structure in which items are ranked according to levels of importance.

HTTP: (Hypertext Transfer Protocol) is the set of rules for transferring files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web.

HTTPS: A web transfer protocol that encrypts and decrypts user page requests as well as the pages that are returned by the Web server. The use of HTTPS protects against eavesdropping and attacks.

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. A group of computers and servers that are connected to each other.

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

Logic (Boolean): Boolean logic deals with the basic operations of truth values: AND, OR, NOT and combinations thereof.

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

Looping: Repetition, using a loop. The action of doing something over and over again.

Lossless: Data compression without loss of information.

Lossy: Data compression in which unnecessary information is discarded.

Memory: Temporary storage used by computing devices.

Model: A representation of (some part of) a problem or a system. (Modeling: the act of creating a model.) Note: This definition differs from that used in science.

Nested loop: A loop within a loop, an inner loop within the body of an outer loop.

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

Operating system: Software that communicates with the hardware and allows other programs to run. An operating system (or "OS") is comprised of system software, or the fundamental files a computer needs to boot up and function. Every desktop computer, tablet, and smartphone includes an operating system that provides basic functionality for the device.

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

Packets: Small chunks of information that have been carefully formed from larger chunks of information.

Pair programming: A technique in which two developers (or students) team together and work on one computer. The terms "driver" and "navigator" are often used for the two roles. In a classroom setting, teachers often specify that students switch roles frequently, or within a specific period of time.

Paradigm (programming): A theory or a group of ideas about how something should be done, made, or thought about. A philosophical or theoretical framework of any kind. Common programming paradigms are object-oriented, functional, imperative, declarative, procedural, logic, and symbolic.

Parallelism: The simultaneous execution on multiple processors of different parts of a program.

Parameter: A special kind of variable used in a procedure to refer to one of the pieces of data provided as input to the procedure. These pieces of data are called arguments. An ordered list of parameters is usually included in the definition of a subroutine so each

time the subroutine is called, its arguments for that call can be assigned to the corresponding parameters. An extra piece of information that you pass to a function to customize it for a specific need.

Pattern matching: Finding similarities between things.

Persistence: Trying again and again, even when something is very hard.

Piracy: The illegal copying, distribution, or use of software.

Procedure: An independent code module that fulfills some concrete task and is referenced within a larger body of source code. This kind of code item can also be called a function or a subroutine. 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. A procedure may also be referred to as a function, subroutine, routine, method, or subprogram.

Processor: The hardware within a computer or device that executes a program. The CPU (central processing unit) is often referred to as the brain of a computer.

Program (n): A set of instructions that the computer executes in order to achieve a particular objective. Program (v): To produce a program by programming. An algorithm that has been coded into something that can be run by a machine.

Programming (v): The craft of analyzing problems and designing, writing, testing, and maintaining programs to solve them. The art of creating a program.

Protocol: The special set of rules that end points in a telecommunication connection use when they communicate. Protocols specify interactions between the communicating entities.

Prototype: An early approximation of a final product or information system, often built for demonstration purposes.

Pseudocode: A detailed yet readable description of what a computer program or algorithm must do, expressed in a formally-styled natural language rather than in a programming language.

Remix: making changes to an existing procedure.

RGB: (red, green, and blue) Refers to a system for representing the colors to be used on a computer display. Red, green, and blue can be combined in various proportions to obtain any color in the visible spectrum.

Routing; router; routing: Establishing the path that data packets traverse from source to destination. A device or software that determines the routing for a data packet.

Run program: Cause the computer to execute the commands written in a program.

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

Servers: Computers that exist only to provide things to others.

Simulate: To imitate the operation of a real world process or system over time.

Simulation: Imitation of the operation of a real world process or system over time.

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

SMTP: A standard protocol for sending emails across the Internet. The communication between mail servers, by default, uses port 25. IMAP: a mail protocol used for accessing email on a remote web server from a local client.

Storage: A place (usually a device) into which data can be entered, in which it can be held, and from which it can be retrieved at a later time. 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 a name, address, or song title. Some functions commonly associated with strings are length, concatenation, and substring.

Structure: The concept of encapsulation without specifying a particular paradigm.

Subroutine: A callable unit of code, a type of procedure.

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. A collection of computing hardware and software integrated for the purpose of accomplishing shared tasks.

Topology: The physical and logical configuration of a network; the arrangement of a network, including its nodes and connecting links. A logical topology details how devices appear connected to the user. A physical topology is how devices 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 computer system.

User: A person for whom a hardware or software product is designed (as distinguished from the developers).

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 the 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. A placeholder for a piece of information that can change. Note: This definition differs from that used in math.

Wearable computing: Miniature electronic devices that are worn under, with or on top of clothing.

Sources for definitions in this glossary:

CAS-Prim: Computing at School. Computing in the national curriculum: A guide for primary teachers (http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf)

Code.org: Creative Commons License (CC BY-NC-SA 4.0) (https://code.org/curriculum/docs/k-5/glossary)

Computer Science Teachers Association: CSTA K–12 Computer Science Standards (2011) https://csta.acm.org/Curriculum/sub/K12Standards.html

FOLDOC: Free On-Line Dictionary of Computing. (http://foldoc.org/)

MA-DLCS: Massachusetts Digital Literacy and Computer Science Standards, Glossary (Draft, December 2015)

NIST/DADS: National Institute of Science and Technology Dictionary of Algorithms and Data Structures. (https://xlinux.nist.gov/dads//)

Techopedia: Techopedia. (https://www.techopedia.com/dictionary)

TechTarget: TechTarget Network. (http://www.techtarget.com/network)

TechTerms: Tech Terms Computer Dictionary. (http://www.techterms.com)

Appendix B-Computer Science Practices

There are seven core practices of computer science. The practices naturally integrate with one another and contain language that intentionally overlaps to illuminate the connections among them. Unlike the core concepts, the practices are not delineated by grade bands. Conversely, like the core concepts, they are meant to build upon each other. (Adapted from: K-12 Computer Science Framework, 2016)

Practices-All practices list skills that students should be able to incorporate by the end of 12th grade
Practice 1. Fostering an Inclusive Computing Culture: Building an inclusive and diverse computing culture requires strategies for incorporating perspectives from people of different genders, ethnicities, and abilities. Incorporating these perspectives involves understanding the personal, ethical, social, economic, and cultural contexts in which people operate. Considering the needs of diverse users during the design process is essential to producing inclusive computational products.
1.1. Include the unique perspectives of others and reflect on one's own perspectives when designing and developing computational products.
1.2. Address the needs of diverse end users during the design process to produce artifacts with broad accessibility and usability.
1.3. Employ self- and peer-advocacy to address bias in interactions, product design, and development methods.
Practice 2. Collaborating Around Computing: Collaborative computing is the process of performing a computational task by working in pairs and on teams. Because it involves asking for the contributions and feedback of others, effective collaboration can lead to better outcomes than working independently. Collaboration requires individuals to navigate and incorporate diverse perspectives, conflicting ideas, disparate skills, and distinct personalities. Students should use collaborative tools to effectively work together and to create complex artifacts.
2.1. Cultivate working relationships with individuals possessing diverse perspectives, skills, and personalities.
2.2. Create team norms, expectations, and equitable workloads to increase efficiency and effectiveness.
2.3. Solicit and incorporate feedback from, and provide constructive feedback to, team members and other stakeholders.
2.4. Evaluate and select technological tools that can be used to collaborate on a project.

Practice 3. Recognizing and Defining Computational Problems: The ability to recognize appropriate and worthwhile opportunities to apply computation is a skill that develops over time and is central to computing. Solving a problem with a computational approach requires defining the problem, breaking it down into parts, and evaluating each part to determine whether a computational solution is appropriate.
3.1. Identify complex, interdisciplinary, real-world problems that can be solved computationally.
3.2. Decompose complex real-world problems into manageable subproblems that could integrate existing solutions or procedures.
3.3. Evaluate whether it is appropriate and feasible to solve a problem computationally.
Practice 4. Developing and Using Abstractions: Abstractions are formed by identifying patterns and extracting common features from specific examples to create generalizations. Using generalized solutions and parts of solutions designed for broad reuse simplifies the development process by managing complexity.
4.1. Extract common features from a set of interrelated processes or complex phenomena.
4.2. Evaluate existing technological functionalities and incorporate them into new designs.
4.3. Create modules and develop points of interaction that can apply to multiple situations and reduce complexity.
4.4. Model phenomena and processes and simulate systems to understand and evaluate potential outcomes.
Practice 5. Creating Computational Artifacts: The process of developing computational artifacts embraces both creative expression and the exploration of ideas to create prototypes and solve computational problems. Students create artifacts that are personally relevant or beneficial to their community and beyond. Computational artifacts can be created by combining and modifying existing artifacts or by developing new artifacts. Examples of computational artifacts include programs, simulations, visualizations, digital animations, robotic systems, and apps.
5.1. Plan the development of a computational artifact using an iterative process that includes reflection on and modification of the plan, taking into account key features, time and resource constraints, and user expectations.
5.2. Create a computational artifact for practical intent, personal expression, or to address a societal issue.
5.3. Modify an existing artifact to improve or customize it.
Practice 6. Testing and Refining Computational Artifacts: Testing and refinement is the deliberate and iterative process of improving a computational artifact. This process includes debugging (identifying and fixing errors) and comparing actual outcomes to intended outcomes. Students also respond to the changing needs and expectations of end users and improve the performance, reliability, usability, and accessibility of artifacts.

6.1. Systematically test computational artifacts by considering all scenarios and using test cases.
6.2. Identify and fix errors using a systematic process.
6.3. Evaluate and refine a computational artifact multiple times to enhance its performance, reliability, usability, and accessibility.
Practice 7. Communicating About Computing: Communication involves personal expression and exchanging ideas with others. In computer science, students communicate with diverse audiences about the use and effects of computation and the appropriateness of computational choices. Students write clear comments, document their work, and communicate their ideas through multiple forms of media. Clear communication includes using precise language and carefully considering possible audiences.
7.1. Select, organize, and interpret large data sets from multiple sources to support a claim.
7.2. Describe, justify, and document computational processes and solutions using appropriate terminology consistent with the intended audience and purpose.
7.3. Articulate ideas responsibly by observing intellectual property rights and giving appropriate attribution.