Arizona Computer Science Standards

June 2018 DRAFT

Arizona Department of Education

High Academic Standards for Students

Contents

Vision Statement 6

Introduction 7

Computer Science Essential Concepts and Subconcepts 9

Computer Science Practices for Students 11

How to Read the Arizona Computer Science Standards 14

Kindergarten 16

- Essential Concept: Computing Systems (CS) 16
- Essential Concept: Networks and the Internet (NI) 17
- Essential Concept: Data and Analysis (DA) 17
- Essential Concept: Algorithms and Programming (AP) 18
- Essential Concept: Impacts of Computing (IC) 20

First Grade 22

- Essential Concept: Computing Systems (CS) 22
- Essential Concept: Networks and the Internet (NI) 23
- Essential Concept: Data and Analysis (DA) ..... 24~~23~~
- Essential Concept: Algorithms and Programming (AP) ..... 25~~24~~
- Essential Concept: Impacts of Computing (IC) ..... 27~~26~~

Second Grade ..... 29~~27~~

- Essential Concept: Computing Systems (CS) ..... 29~~27~~
- Essential Concept: Networks and the Internet (NI) ..... 30~~28~~
- Essential Concept: Data and Analysis (DA) ..... 31~~29~~

Essential Concept: Algorithms and Programming (AP) ..... 32~~30~~
Essential Concept: Impacts of Computing (IC) ..... 34~~31~~
Third Grade ..... 35~~33~~
Essential Concept: Computing Systems (CS) ..... 35~~33~~
Essential Concept: Networks and the Internet (NI) ..... 36~~34~~
Essential Concept: Data and Analysis (DA) ..... 36~~34~~
Essential Concept: Algorithms and Programming (AP) ..... 37~~35~~
Essential Concept: Impacts of Computing (IC) ..... 40~~37~~
Fourth Grade ..... 42~~39~~
Essential Concept: Computing Systems (CS) ..... 42~~39~~
Essential Concept: Networks and the Internet (NI) ..... 43~~40~~
Essential Concept: Data and Analysis (DA) ..... 44~~40~~
Essential Concept: Algorithms and Programming (AP) ..... 44~~41~~
Essential Concept: Impacts of Computing (IC) ..... 47~~43~~
Fifth Grade ..... 49~~45~~
Essential Concept: Computing Systems (CS) ..... 49~~45~~
Essential Concept: Networks and the Internet (NI) ..... 50~~46~~
Essential Concept: Data and Analysis (DA) ..... 51~~46~~
Essential Concept: Algorithms and Programming (AP) ..... 52~~47~~
Essential Concept: Impacts of Computing (IC) ..... 54~~49~~
Sixth Grade ..... 57~~51~~
Essential Concept: Computing Systems (CS) ..... 57~~51~~

Essential Concept: Networks and the Internet (NI) ..... 58~~52~~
Essential Concept: Data and Analysis (DA) ..... 58~~52~~
Essential Concept: Algorithms and Programming (AP) ..... 59~~53~~
Essential Concept: Impacts of Computing (IC) ..... 61~~55~~
Seventh Grade ..... 63~~57~~
Essential Concept: Computing Systems (CS) ..... 63~~57~~
Essential Concept: Networks and the Internet (NI) ..... 64~~58~~
Essential Concept: Data and Analysis (DA) ..... 65~~58~~
Essential Concept: Algorithms and Programming (AP) ..... 65~~59~~
Essential Concept: Impacts of Computing (IC) ..... 68~~61~~
Eighth Grade ..... 70~~63~~
Essential Concept: Computing Systems (CS) ..... 70~~63~~
Essential Concept: Networks and the Internet (NI) ..... 71~~64~~
Essential Concept: Data and Analysis (DA) ..... 72~~65~~
Essential Concept: Algorithms and Programming (AP) ..... 73~~66~~
Essential Concept: Impacts of Computing (IC) ..... 75~~68~~
High School ..... 77~~70~~
Essential Concept: Computing Systems (CS) ..... 77~~70~~
Essential Concept: Networks and the Internet (NI) ..... 78~~71~~
Essential Concept: Data and Analysis (DA) ..... 79~~72~~
Essential Concept: Algorithms and Programming (AP) ..... 80~~73~~
Essential Concept: Impacts of Computing (IC) ..... 82~~75~~

Computer Science Glossary ..... 85~~77~~
References: ..... 92~~83~~

Vision Statement

All K-12 students are expected *to* develop a foundation of computer science knowledge and learn new approaches to problem solving and critical thinking to become ~~both~~ ethical, responsible users and creators of computing technology ~~to ensure that all~~. S~~s~~tudents ~~have~~ will develop ~~the~~ basic knowledge that will allow them to productively participate in the world and make well informed decisions about their lives.

Understanding problems, their potential solutions, and the technologies, techniques, and resources needed to solve them are critical for citizens of the 21st century. Toward that end, the state of Arizona has created these standards to further this understanding. The standards will allow students to develop a foundation of computer science knowledge and learn new approaches to problem solving that capture~~s~~ the power of computational thinking ~~to~~ and enable them to become ~~both~~ 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
- Better understand the role of computing in the world around them
- Learn, perform, and express themselves critically in a variety of subjects and interests

Equity is ~~of~~ also a top priority ~~also~~. ~~This~~ An equity focus ~~will~~ ensure~~s~~ that all students can engage in the content of the computer science standards with sufficient flexibility to allow ~~everyone~~ them to demonstrate proficiency in multiple ways.

Introduction

Understanding problems and their potential solutions~~,~~ (including the technologies, techniques, and resources needed to solve them) are critical for citizens of the 21st century. The state of Arizona has created computer science standards to further this understanding. These standards will allow students to develop a foundation of computer science knowledge ~~by~~ and to learn~~ing~~ new approaches to problem solving that capture the power of computational thinking and enable students to become both 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
- Better understand the role of computing in the world around them
- Learn, perform, and express themselves critically in a variety of subjects and interests

With efforts to broaden participation, equity is an important factor across the discipline of computer science. Educators and students who engage with these standards are encouraged to actively seek collaboration with those who have different perspectives and backgrounds. This attention to equity serves to strengthen individual communities and prepare students to be informed and responsible users of technology. Furthermore, the Arizona Computer Science Standards provide flexibility to allow all students to demonstrate proficiency in multiple ways, thus providing a maximum opportunity for engagement in computer science.

The Arizona Computer Science Standards integrate computer literacy, educational technology, digital citizenship, information technology, and computer science. Computer literacy, educational technology, digital citizenship, and information technology are concepts that students are also exposed to in the Computer Technology Standards of Learning. In many ways, instruction in the computer science standards will compliment and ~~expound upon, at a deeper level,~~expand and deepen the concepts and skills covered within the Computer Technology Standards. However, there are distinct differences between computer technology and computer science. 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 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.

The previously listed concepts tend to focus more on the using computer technologies as opposed to understanding why they work and how to create those technologies (K-12 Computer Science Framework, 2016).

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.

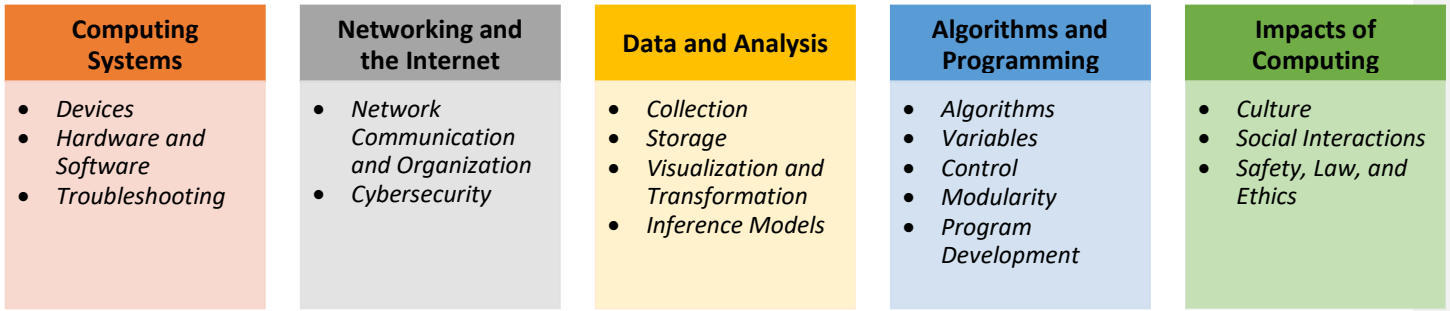
- **Data and Analysis:** This involves the data that exist and the computing systems that exist to process that data. Data is collected and stored so that it can be analyzed to better understand the world and make more accurate predictions. The amount of digital data generated in the world is rapidly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

Essential 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 essential 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~~ acquire skills for developing and using abstractions. Identifying patterns and extracting common features from specific examples enables students to create generalizations and to 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.

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 capture~~s~~ the power of computational thinking, enabling them ~~to~~ 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: Computer science practices

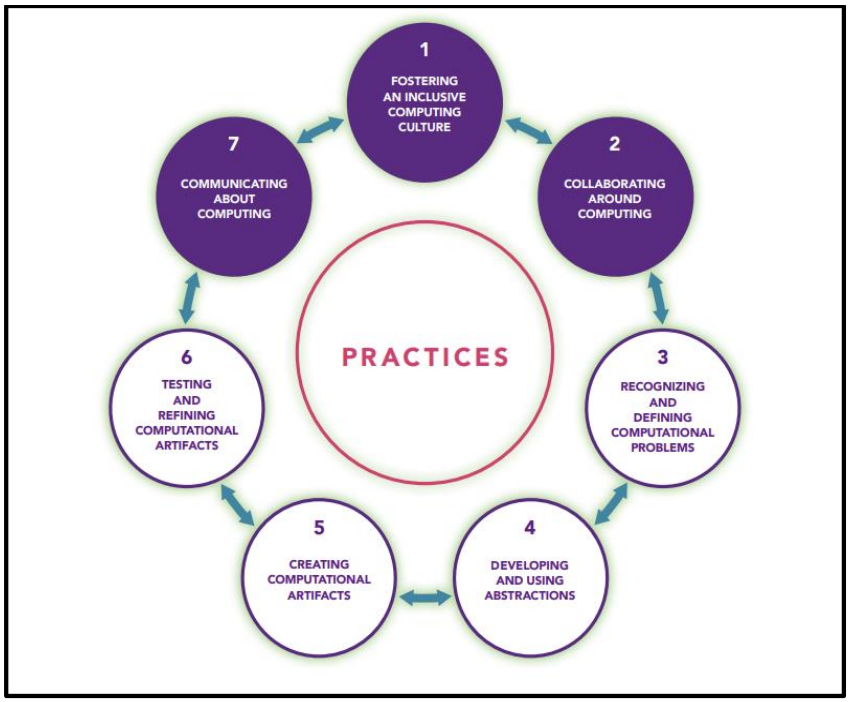
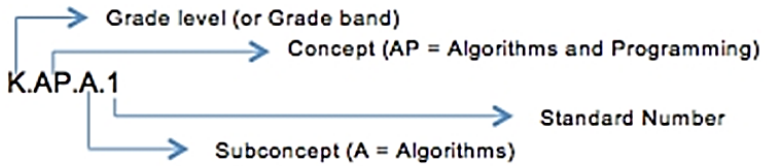


Figure 2: Practices- K-12 Computer Science Framework. (2016)

How to Read the Arizona Computer Science Standards

The Arizona K-12 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 essential concepts. These essential concepts include Computing Systems, Networks and the Internet, Data and Analysis, Algorithms and Programming, and Impacts of Computing. Within each essential concept there may be two to five subconcepts represented. Each standard ties back to a practice that every computationally literate student will engage in as they develop computational artifacts (see Figure 3). Each standard will list (in order): The grade level, the essential concept being covered, the subconcept being covered and finally, the number of the standard. The finished product will look like the example below in **Figure 3**. This example shows two different subconcepts under the essential concept of Algorithms and Programing at the Kindergarten level.

Figure 3. Example of subconcepts listed under an essential concept



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. Other examples of algorithms include making simple foods, navigating a classroom, and daily routines like brushing teeth. 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.* *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

By the end of Kindergarten, students 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. Students will be able to explain the importance of password protection and discuss how computer networks can connect people globally. Students will use digital devices to collect, modify, store, and display data to make inferences or predictions. With teacher assistance, the computer literate student will begin to develop an understanding of how to model and develop algorithms and programs. Students will be introduced to the impacts of computing.

Essential 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: 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: Communicating about Computing: 7.2
Subconcept: Troubleshooting (T)	

Commented [MOU1]: This is appropriate for pre-literate children.

Commented [MOU2]: How is this fostering inclusivity?

K.CS.T.1	**Discuss basic hardware and software problems.** *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 (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: Testing and Refining Computational Artifacts, Communicating About Computing: 6.2, 7.3

Essential 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. 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.* *Practice(s):* Communicating About Computing: 7.2
Subconcept: Network, Communication, and Organization (NCO)	
K.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.* *Practice(s): Communicating About Computing: 7.3*

Essential 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.* *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 or the averages for colors in multiple bags 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. Students could analyze graphs of temperatures taken at the beginning of the school day and end of the school day, identify the patterns of when temperatures rise and fall, and predict if they think the temperature will rise or fall at a particular time of the day, based on the pattern observed.* *Practice(s):* Developing and Using Abstractions: 4.1

Essential 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. Other examples of algorithms include making simple foods, navigating a classroom, and daily routines like brushing teeth. 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.* *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 desired 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.* 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 a task into multiple~~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: 3.1

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 their program will do. Students at this stage may complete the planning process with help from their teachers.* 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.* 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 programs they create and the choices that they made when creating programs. This could be done using coding journals, discussions with a teacher, class presentations, or blogs.* *Practice(s)*: Communicating About Computing: 7.2

Essential Concept: Impacts of Computing (IC)

Subconcept: Culture (C)	
K.IC.C.1	**Discuss differences in 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 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: 7.2
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. 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)	
K.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 interactions in the world, such as "stranger danger," apply to online environments as well.* Practice(s): Communicating About Computing:7.2

Commented [MOU3]: This standard doesn't make sense with pre-literate children who are unlikely to be communicating online. Better to focus on basic concepts of respect and sharing for other users.

First Grade

By the end of first grade, students will recognize user needs and preferences while ~~utilizing devices~~using hardware and software to perform basic computer operations, hardware, software, and apply basic troubleshooting strategies. 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. The computer literate student 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.

Essential Concept: Computing Systems (CS)

Subconcept: Devices (D)	
1.CS.D.1	**With teacher guidance, select and operate appropriate device and software to perform a task and recognize that users have different needs and preferences for the technology they use.** *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, 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 compare and discuss preferences for software with the same primary functionality. Students could compare different web browsers or word processing, presentation, or drawing programs.* Practice: 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: Communicating about Computing: 7.2

Commented [MOU4]: This is beyond the capability for pre-literate children.

Subconcept: Troubleshooting (T)	
1.CS.T.1	**Identify basic hardware and software 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, a device will not turn on, the sound does not work, etc.).* Practice: 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.** *Ideally, students would 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. These are, however, not specified in the standard, because these problems may not occur.* Practice: Testing and Refining Computational Artifacts: 6.2

Essential 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. 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.* *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.*

	Practice(s): Communicating About Computing: 7.2

Essential 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. *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 or the averages for colors in multiple bags 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. Students could analyze graphs of temperatures taken at the beginning of the school day and end of the school day, identify the patterns of when*

	temperatures rise and fall, and predict if they think the temperature will rise or fall at a particular time of the day, based on the pattern observed. *Practice(s):* Developing and Using Abstractions: 4.4

Essential 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. Other examples of algorithms include making simple foods, navigating a classroom, and daily routines like brushing teeth. 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.* *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 desired task~~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.*

	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 a task into multiple ~~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: 3.1
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 program will do. Students at this stage may complete the planning process with help from their teachers.* *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.* *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 programs they create and the choices that they made when creating programs. This could be done using coding journals, discussions with a teacher, class presentations, or blogs.*

Commented [MOU5]: Suggest removing as blog posts and websites usually do not involve oral representations.

	Practice(s): Communicating About Computing: 7.2

Essential Concept: Impacts of Computing (IC)

Subconcept: Culture (C)	
1.IC.C.1	**Discuss differences in 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 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 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 interactions in the world, such as "stranger danger," apply to online environments as well.* *Practice(s):* Communicating About Computing: 7.2

Commented [MOU6]: This standard doesn't make sense with pre-literate children who are unlikely to be communicating online. Better to focus on basic concepts of respect and sharing for other users.

Second Grade

By the end of second grade, students will be able to select the appropriate device and software to perform specific tasks. They 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. Students will independently collect, transform, and display data using digital devices; they can store, copy, search, retrieve, modify, and delete information using a computing device. Students will be able to create and follow algorithms and programs using loops and variables to solve a problem. The computer science literate student will decompose steps into simpler tasks and give credit to ideas and creations of others.

Essential 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 compare and discuss preferences for software with the same primary functionality. Students could compare different web browsers or word processing, presentation, or drawing programs.* *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 and software 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. ~~Understand that~~In ~~i~~nput, processing, and output of information is the central purpose of computing.* *Practice(s):* Communicating about Computing : 7.2
Subconcept: Troubleshooting (T)	

Commented [MOU7]: It is difficult to assess "recognize". Suggest "Describe different preferences and needs users may have when selecting and operating computing devices."

2.CS.T.1	**Explain basic hardware and software 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, a device will not turn on, the sound does not work, etc.).* *Practice(s):* Communicating About Computing: 7.2
2.CS.T.2	**With teacher guidance, use basic troubleshooting strategies.** *Ideally, students would 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. These are, however, not specified in the standard, because these problems may not occur.* *Practice(s):* Testing and Refining Computational Artifacts: 6.2

Essential 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~~ Students 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.* *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.* *Practice(s): Communicating About Computing: 7.2*

Essential Concept: Data and Analysis (DA)

Subconcept: Collection, Visualization and Transformation (CVT)	
2.DA.CVT.1	**Collect and transform data using digital devices~~; D~~ and 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. *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 or the averages for colors in multiple bags 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. Students could analyze graphs of temperatures taken at the beginning of the school day and end of the school day, identify the patterns of when temperatures rise and fall, and predict if they think the temperature will rise or fall at a particular time of the day, based on the pattern observed.* *Practice(s):* Developing and Using Abstractions: 4.3

Essential 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. Other examples of algorithms include making simple foods, navigating a classroom, and daily routines like brushing teeth. 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.* *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 desired ~~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.* *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 a task into multiple~~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 their program will do. Students at this stage may complete the planning process with help from their teachers.* *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. 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
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.* *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 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, discussions with a teacher, class presentations, or blogs.* *Practice(s):* Communicating About Computing: 7.2

Commented [MOU8]: Suggest removing as blog posts and websites usually do not involve oral representations.

Essential Concept: Impacts of Computing (IC)

Subconcept: Culture (C)	
2.IC.C.1	**Compare 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 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 interactions in the world, such as "stranger danger," apply to online environments as well.* *Practice(s):* Communicating About Computing: 7.2

Third Grade

By the end of third grade, students will explore a variety of computing devices and tools to further develop their computational thinking and problem-solving skills. They will be able to collaborate locally and globally with peers. Students will plan, make predictions, solve problems, and draw conclusions about data, programs, and computational artifacts. The computer science literate student will learn the importance of protecting personal information and respecting the rights of others.

Essential Concept: Computing Systems (CS)

Subconcept: Devices (D)	
3.CS.D.1	**Identify how internal and external parts of computing devices function to form a system within a single device and hardware that connects to the device to extend capability.** *Keyboard input or a mouse click could cause an action to happen or information to be displayed on a screen; this could only happen because the computer has a processor to evaluate what is happening externally and produce corresponding responses. Students should describe how devices and components interact using correct terminology.* *Practice(s): Communicating About Computing, Recognizing and Defining Computational Problems: 7.2, 3.2*
Subconcept: Hardware and Software (HS)	
3.CS.HS.1	**Recognize that hardware and software communicate in a special language that the computing system can understand.** *Explanation: Computing systems convert instructions, such as "print," "save," or "crop," into a special language that the computer can understand.* *Practice(s): Communicating About Computing: 7.2*
3.CS.HS.2	**Recognize that hardware can only accomplish the specific tasks the software is designed to accomplish.** *For example, a camera can take picture because the camera software allows it to ~~take pictures~~do so.* *Practice(s): Communicating About Computing: 7.2*
Subconcept: Troubleshooting (T)	

Commented [MOU9]: You use the "Explanation:" precursor for some standards and not for others. It really isn't needed, so I suggest removing it from this and all following standards where it appears.

3.CS.T.1	**Identify and ~~utilize~~ use common troubleshooting strategies to solve simple hardware and software problems.** *Although computing systems may vary, common troubleshooting strategies ~~can be used on them,~~ such as checking connections and power or swapping a working part in place of a potentially defective part can be used to restore functionality. Restarting a computer (rebooting) is commonly effective because it resets the machine.~~Rebooting a machine is commonly effective because it resets the computer.~~ Because computing devices are composed of an interconnected system of hardware and software, troubleshooting strategies may need to address both.* *Practice(s): Communicating About Computing: 7.2*

Essential Concept: Networks and the Internet (NI)

Subconcept: Cybersecurity (C)	
3.NI.C.1	**Identify real-world cybersecurity problems and how personal information can be protected.** *Just as we protect our personal property online, we also need to protect our devices and the information stored on them. Information can be protected using various security measures. These measures can be physical and/or digital. For example, discussion topics could be based on current events related to cybersecurity or topics that are applicable to students and the programs/devices they use.* *Practice(s): Communicating about Computing, Recognizing and Defining Computational Problems: 7.1, 3.1*
Subconcept: Network, Communication, and Organization (NCO)	
3.NI. NCO.1	**Model how information ~~flows in a physical or wireless path to travel to be sent and received~~ is sent and received via a physical or wireless path.** *There are physical paths for communicating information, such as Ethernet cables, and wireless paths, such as Wi-Fi. Often, information travels on a combination of physical and wireless paths; for example, wireless paths originate from a physical connection point.* *Practice(s): Developing and Using Abstractions: 4.3*

Essential Concept: Data and Analysis (DA)

Subconcept: Collection, Visualization and Transformation (CVT)

3.DA.CVT.1	**Select tools from a specified list to collect, organize, and present data visually to highlight relationships and support a claim.** *Tools are chosen based upon the type of measurement they use as well as the type of data people wish to observe. Organizing data can make interpreting and communicating it to others easier. Data points can be clustered by a number of commonalities.* *Practice(s): Developing and Using Abstractions, Creating Computational Artifacts: 4.1, 5.1*
Subconcept: Storage (S)	
3.DA.S.1	**Recognize different file extensions.** *Music, images, video, and text require different amounts of storage. Video will often require more storage than music or images alone because video combines both.* *Practice(s): Communicating About Computing: 7.2*
Subconcept: Inference and Models (IM)	
3.DA.IM.1	**Use a computational tool to draw conclusions, make predictions, and answer questions utilizing a specified data set.** *People use data to highlight or predict outcomes. Basing inferences or predictions on data does not guarantee their accuracy; the data must be relevant and of sufficient quantity.* *Practice(s): Communicating about Computing, Collaborate around Computing: 7.2, 2.4*

Commented [MOU10]: There is nothing in this standard that support collaboration.

Essential Concept: Algorithms and Programming (AP)

Subconcept: Algorithms (A)	
3.AP.A.1	**Recognize and compare multiple algorithms for the same task and determine which are appropriate.** *Different algorithms can achieve the same result, though sometimes one algorithm might be most appropriate for a specific ~~situation~~task. Students should be able to look at different ways to solve the same task and decide which would be the best solution. For example, students might compare algorithms that describe how to get ready for school or how to tie their shoes. Students could use a map and plan multiple algorithms to get from one point to another. Another example might be to write different algorithms to draw a regular polygon and determine which algorithm would be the easiest to modify or repurpose to draw a different polygon.* *Practice(s):* Developing and Using Abstractions, 4.4
Subconcept: Variables (V)	

3.AP.V.1	**Create programs that use variables to store and modify data.** *Variables are used to store and modify data. At this level, understanding how to use variables is sufficient. Data types vary by programming language, but many have types for numbers and text. For examples, students may use mathematical operations to add to the score of a game or subtract from the number of lives in a game. The use of a variable as a countdown timer is another example. Programs can imply either digital or ~~paper-based~~paper-based designs.* *Practice(s):* Creating Computational Artifacts: 5.2
Subconcept: Control (C)	
3.AP.C.1	**Create programs that include sequences, events, loops, and/or conditionals.** *Control structures specify the order (sequence) in which instructions are executed within a program and can be combined to support the creation of more complex programs. For example, if dialogue is not sequenced correctly when programming a simple animated story, the story will not make sense. If the commands to program a robot are not in the correct order, the robot will not complete the desired task~~task desired~~. Events allow portions of a program to run based on a specific action. For example, students could write a program to explain the water cycle and when a specific component is clicked (event), the program would show information about that part of the water cycle. Loops allow for the repetition of a sequence of code multiple times. For example, in a program that produces an animation about a famous historical character, students could use a loop to have the character walk across the screen as they introduce themselves. Conditionals allow for the execution of a portion of code in a program when a certain condition is true. For example, students could write a math game that asks multiplication fact questions and then uses a conditional to check whether or not the answer that was entered is correct.* Practice(s): Creating Computational Artifacts: 5.2
Subconcept: Modularity (M)	
3.AP.M.1	**Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.** *Decomposition is the act of breaking down a task into multiple~~tasks~~ ~~into~~ simpler tasks. Decomposition also enables different people to work on different parts at the same time. For example, students could create an animation by separating a story into different scenes. For each scene, they would select a background, place characters, and program actions.* Practice(s): Recognizing and Defining Computational Problems: 3.2
Subconcept: Program Development (PD)	

3.AP.PD.1	**With teacher guidance, use an iterative process to plan the development of a program by including others' perspectives and considering user preferences.** *Planning is an important part of the iterative process of program development. Students outline key features, time and resource constraints, and user expectations. Students should document the plan as, for example, a storyboard, flowchart, pseudocode, or story map.* Practice(s): Fostering an Inclusive Computing Culture, Creating Computational Artifacts: 1.1, 5.1
3.AP.PD.2	**Observe intellectual property rights and give appropriate attribution when creating or remixing programs.** *Intellectual property rights can vary by country but copyright laws give the creator of a work a set of rights that prevents others from copying the work and using it in ways that ~~they~~ the creator may not like. Students should identify instances of remixing, when ideas are borrowed and iterated upon, and credit the original creator. Students should also consider common licenses that place limitations or restrictions on the use of computational artifacts, such as images and music downloaded from the Internet. At this stage, attribution should be written in the format required by the teacher and should always be included on any programs shared online.* Practice(s): Creating Computational Artifacts, Communicating About Computing: 5.2, 7.3
3.AP.PD.3	**Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.** *As students develop programs they should continuously test those programs to see that they do what was expected and fix (debug)~~,~~ any errors. Students should also be able to successfully debug simple errors in programs created by others.* Practice(s): Testing and Refining Computational Artifacts: 6.1, 6.2
3.AP.PD.4	**Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development.** *Collaborative computing is the process of performing a computational task by working in pairs or on teams. Because it involves asking for the contributions and feedback of others, effective collaboration can lead to better outcomes than working independently. Students should take turns in different roles during program development, such as note taker, facilitator, program tester, or "driver" of the computer.* Practice(s): Collaborating Around Computing: 2.2
3.AP.PD.5	**Describe choices made during program development using code comments, presentations, and/or demonstrations.** *People communicate about their code to help others understand and use their programs. Another purpose of communicating one's design choices is to show an understanding of one's work. These explanations could manifest themselves as in-line code comments for collaborators and assessors, or as part of a summative presentation, such as a code walk-through or coding journal.* Practice(s): Communicating About Computing: 7.2

Essential Concept: Impacts of Computing (IC)

Subconcept: Culture (C)	
3.IC.C.1	**Identify computing technologies that have changed the world.** *New computing technology is created and existing technologies are modified for many reasons, including to increase their benefits, decrease their risks, and meet societal needs. Students, with guidance from their teacher, should discuss topics that relate to the history of technology and the changes in the world due to technology. Topics could be based on current news content in areas~~,~~ such as robotics, wireless Internet, mobile computing devices, GPS systems, wearable computing, or how social media has influenced social and political changes.* Practice(s): Recognizing and Defining Computational Problems: 3.1
3.IC.C.2	**With teacher guidance, brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users.** *The development and modification of computing technology are driven by people's needs and wants and can affect groups differently. Anticipating the needs and wants of diverse end users requires students to purposefully consider potential perspectives of users with different backgrounds, ability levels, points of view, and disabilities. For example, students may consider using both speech and text when they wish to convey information in a game. They may also wish to vary the types of programs they create, knowing that not everyone shares ~~share~~ their own tastes.* Practice(s): Fostering an Inclusive Computing Culture: 1.2
Subconcept: Social Interactions (SI)	
3.IC.SI.1	**Seek opportunities for local and global collaboration to facilitate communication and innovation.** *Computing influences many social institutions such as family, education, religion, and the economy. People can work in different places and at different times to collaborate and share ideas when they use technologies that reach across the globe. Computing provides the possibility for collaboration and sharing of ideas and allows the benefit of diverse perspectives. These social interactions affect how local and global groups interact with each other, and alternatively, these interactions can change the nature of groups. For example, a class can discuss ideas in the same class, school, or in another state or nation through interactive webinars.* Practice(s): Fostering an Inclusive Computing Culture: 1.1
Subconcept: Safety, Law, and Ethics (SLE)	

3.IC.SLE.1	**Use public domain or creative commons media, and refrain from copying or using material created by others without permission.** *Ethical complications arise from the opportunities provided by computing. The ease of sending and receiving copies of media, such as video, photos, and music , on the Internet , such as video, photos, and music, creates the opportunity for unauthorized use, such as online piracy and disregard of copyrights. Students should consider the licenses on for the computational artifacts that they wish to use. For example, the license on a downloaded image or audio file may have restrictions that prohibit modification, require attribution, or prohibit use entirely.* Practice(s): Communicating About Computing: 7.3

Fourth Grade

By the end of fourth grade, students refine their skills as they construct programs and ~~utilize~~ use algorithms to accomplish a task. They learn to analyze a variety of hardware and software tools; and will further develop their computational thinking and problem-solving skills. Working independently and collaboratively, students decompose larger problems into smaller tasks. Students understand that responsible computing use includes ~~by~~ protecting personal information and respecting the rights of others~~, students continue to be responsible users of technology~~. The computer science literate student begins to think about the impacts of computing and computing devices.

Essential Concept: Computing Systems (CS)

Subconcept: Devices (D)	
4.CS.D.1	**With teacher guidance model how internal and external parts of computing ~~devices function to form a system between~~connect multiple devices in a computing system.** *Computing devices may be connected to other devices or components to extend their capabilities, such as sensing and sending information. Connections can take many forms, such as physical or wireless. Together, devices and components form a system of interdependent parts that interact for a common purpose* *Practice(s): Communicating About Computing, Recognizing and Defining Computational Problems, Creating Computational Artifacts: 7.3, 3.1, 5.2*
Subconcept: Hardware and Software (HS)	
4.CS.HS.1	**Recognize that bits serve as the basic unit of data in computing systems and can represent a variety of information.** *Hardware and software communicate in binary digits commonly represented in 0s and 1s* *Practice(s): Communicating About Computing: 7.2*
4.CS.HS.2	**Recognize that a single piece of hardware can accomplish different tasks depending on ~~the~~ its software.** *For example, a photo filter application (software) works with a camera (hardware) to produce a variety of effects that change the appearance of an image. This image is transmitted and stored as bits, or binary digits, which are commonly represented as 0s and 1s. All information, including instructions, is encoded as bits.* *Practice(s): Practice(s): Communicating About Computing: 7.2*

Subconcept: Troubleshooting (T)	
4.CS.T.1	**Develop a model with teacher guidance common troubleshooting strategies to solve simple hardware and software problems.** *Although computing systems may vary, common troubleshooting strategies such as checking connections and power or swapping a working part in place of a potentially defective part can be used to restore functionality~~can be used on them, such as checking connections and power or swapping a working part in place of a potentially defective part~~. Restarting a computer (rebooting)~~Rebooting a machine~~ is commonly effective because it resets the ~~computer~~machine. Because computing devices are composed of an interconnected system of hardware and software, troubleshooting strategies may need to address both.* *Practice(s): Recognizing and Defining Computational Problems, Collaborating Around Computing: 3.1, 2.4*

Essential Concept: Networks and the Internet (NI)

Subconcept: Cybersecurity (C)	
4.NI.C.1	**Discuss real-world cybersecurity problems and how personal information can be protected.** *Just as we protect our personal property on line, we also need to protect our devices and the information stored on them. Information can be protected using various security measures. These measures can be physical and/or digital. For example, discussion topics could be based on current events related to cybersecurity or topics that are applicable to students and the programs/devices they use.* *Practice(s): Communicating about Computing, Recognizing and Defining Computational Problems: 7.2, 3.3*
Subconcept: Network, Communication, and Organization (NCO)	
4.NI. NCO.1	**Model how information is broken down into smaller pieces, transmitted as packets through multiple devices over networks and the Internet, and reassembled at the destination.** *There are physical paths for communicating information, such as Ethernet cables, and wireless paths, such as Wi-Fi. Often, information travels on a combination of physical and wireless paths; for example, wireless paths originate from a physical connection point. Information is broken down into smaller pieces called packets, which are sent independently and reassembled at the destination. Routers and switches are used to properly send packets across paths to their destinations.* *Practice(s): Developing and Using Abstractions: 4.4*

Essential Concept: Data and Analysis (DA)

Subconcept: Collection, Visualization and Transformation (CVT)	
4.DA.CVT.1	**Independently select tools to collect, organize, and present data visually to highlight relationships and support a claim.** *Tools are chosen based upon the type of measurement they use as well as the type of data people wish to observe. Organizing data can make interpreting and communicating it to others easier. Data points can be clustered by a number of commonalities.* *Practice(s): Developing and Using Abstractions, Creating Computational Artifacts: 4.1, 5.1*
Subconcept: Storage (S)	
4.DA.S.1	**Recognize different file extensions and how they are stored on a computing device.** *Music, images, video, and text require different amounts of storage. Video will often require more storage than music or images alone because video combines both. ~~Define cloud storage.~~ For example, two pictures of the same object can require different amounts of storage based upon their resolution. Define cloud storage.* *Practice(s): Communicating About Computing: 7.2*
Subconcept: Inference and Models (IM)	
4.DA.IM.1	**Use a computational tool to manipulate data to draw conclusions, make predictions, and answer questions.** *People use data to highlight or propose cause-and-effect relationships and predict outcomes. Basing inferences or predictions on data does not guarantee their accuracy; the data must be relevant and of sufficient quantity.* *Practice(s): Communicating about Computing, Creating Computational Artifacts, Collaborate around Computing: 7.2, 5.2, 2.4*

Commented [MOU11]: Collaboration is not explicit in this standard.

Essential Concept: Algorithms and Programming (AP)

Subconcept: Algorithms (A)	
4.AP.A.1	**Compare and refine multiple algorithms for the same task and determine which is the most appropriate.** Different algorithms can achieve the same result, though sometimes one algorithm might be most appropriate for a specific situation. Students should be able to look at different ways to solve the same task and decide which would be the best solution. For example, students might compare algorithms that describe how to get ready for school or how to tie their shoes. Students could use a map and plan multiple algorithms to get from one point to another. They could look at routes suggested by mapping software and change the route to something that would be better, based on which route is shortest or fastest or would avoid a

Commented [MOU12]: Do you really mean "appropriate" or do you mean "effective"?

	problem. Another example might be to write different algorithms to draw a regular polygon and determine which algorithm would be the easiest to modify or repurpose to draw a different polygon. *Practice(s):* Testing and Refining Computational Artifacts, Recognizing and Defining Computational Problems: 6.3
Subconcept: Variables (V)	
4.AP.V.1	**Create programs that use variables to store and modify data, recognizing that the data type determines the values that can be stored and the operations that can be performed on the data.** *Variables are used to store and modify data. At this level, understanding how to use variables is sufficient, without a fuller understanding of the technical aspects of variables (such as identifiers and memory locations). Data types vary by programming language, but many have types for numbers and text. For examples, students may use mathematical operations to add to the score of a game or subtract from the number of lives in a game. The use of a variable as a countdown timer is another example. Programs can imply either digital or paper-based ~~paper based~~ designs.* *Practice(s):* Creating Computational Artifacts: 5.2
Subconcept: Control (C)	
4.AP.C.1	**Create programs that include sequences, events, loops, and/or conditionals.** *Control structures specify the order (sequence) in which instructions are executed within a program and can be combined to support the creation of more complex programs. For example, if dialogue is not sequenced correctly when programming a simple animated story, the story will not make sense. If the commands to program a robot are not in the correct order, the robot will not complete the desired task~~task desired~~. Events allow portions of a program to run based on a specific action. For example, students could write a program to explain the water cycle and when a specific component is clicked (event), the program would show information about that part of the water cycle. Loops allow for the repetition of a sequence of code multiple times. For example, in a program that produces an animation about a famous historical character, students could use a loop to have the character walk across the screen as they introduce themselves. Conditionals allow for the execution of a portion of code in a program when a certain condition is true. For example, students could write a math game that asks multiplication fact questions and then uses a conditional to check whether or not the answer that was entered is correct.* Practice(s): Creating Computational Artifacts: 5.2
Subconcept: Modularity (M)	

4.AP.M.1	**Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.** *Decomposition is the act of breaking down a task into multiple ~~tasks into~~ simpler tasks. Decomposition also enables different people to work on different parts at the same time. For example, students could create an animation by separating a story into different scenes. For each scene, they would select a background, place characters, and program actions.* Practice(s): Recognizing and Defining Computational Problems: 3.2
4.AP.M.2	**Modify, remix, or incorporate portions of an existing program into one's own work to add more advanced features.** *Programs can be broken down into smaller parts, which can be incorporated into new or existing programs. For example, students could modify prewritten code from a single-player game to create a two-player game with slightly different rules, remix and add another scene to an animated story, use code to make a ball bounce from another program in a new basketball game, or modify an image created by another student.* Practice(s): Creating Computational Artifacts: 5.3
Subconcept: Program Development (PD)	
4.AP.PD.1	**Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences.** *Planning is an important part of the iterative process of program development. Students outline key features, time and resource constraints, and user expectations. Students should document the plan as, for example, a storyboard, flowchart, pseudocode, or story map.* Practice(s): Fostering an Inclusive Computing Culture, Creating Computational Artifacts: 1.1, 5.1
4.AP.PD.2	**Observe intellectual property rights and give appropriate attribution when creating or remixing programs.** *Intellectual property rights can vary by country but copyright laws give the creator of a work a set of rights that prevents others from copying the work and using it in ways that the creator ~~they~~ may not like. Students should identify instances of remixing, when ideas are borrowed and iterated upon, and credit the original creator. Students should also consider common licenses that place limitations or restrictions on the use of computational artifacts, such as images and music downloaded from the Internet. At this stage, attribution should be written in the format required by the teacher and should always be included on any programs shared online.* Practice(s): Creating Computational Artifacts, Communicating About Computing: 5.2, 7.3
4.AP.PD.3	**Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.** *As students develop programs, they should continuously test those programs to see that they do what was expected and fix (debug) ~~,~~ any errors. Students should also be able to successfully debug simple errors in programs created by others.* Practice(s): Testing and Refining Computational Artifacts: 6.1, 6.2

4.AP.PD.4	**Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development.** *Collaborative computing is the process of performing a computational task by working in pairs or on teams. Because it involves asking for the contributions and feedback of others, effective collaboration can lead to better outcomes than working independently. Students should take turns in different roles during program development, such as note taker, facilitator, program tester, or "driver" of the computer.* Practice(s): Collaborating Around Computing: 2.2
4.AP.PD.5	**Describe choices made during program development using code comments, presentations, and/or demonstrations.** *People communicate about their code to help others understand and use their programs. Another purpose of communicating one's design choices is to show an understanding of one's work. These explanations could manifest themselves as in-line code comments for collaborators and assessors, or as part of a summative presentation, such as a code walk-through or coding journal.* Practice(s): Communicating About Computing: 7.2

Essential Concept: Impacts of Computing (IC)

Subconcept: Culture (C)	
4.IC.C.1	**Identify and discuss computing technologies that have changed the world, and express how those technologies influence, and are influenced by, cultural practices.** *New computing technology is created and existing technologies are modified for many reasons, including to order to increase their benefits, decrease their risks, and meet societal needs. Students, with guidance from their teacher, should discuss topics that relate to the history of technology and the changes in the world due to technology. Topics could be based on current news content~~,~~ in areas such as robotics, wireless Internet, mobile computing devices, GPS systems, wearable computing, or how social media has influenced social and political changes.* Practice(s): Recognizing and Defining Computational Problems: 3.1
4.IC.C.2	**Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users.** *The development and modification of computing technology are driven by people's needs and wants and can affect groups differently. Anticipating the needs and wants of diverse end users requires students to purposefully consider potential perspectives of users with different backgrounds, ability levels, points of view, and disabilities. For example, students may consider using both speech and text when they wish to convey information in a game. They may also wish to vary the types of programs they create, knowing that not everyone shares ~~share~~ their own tastes.*

	Practice(s): Fostering an Inclusive Computing Culture: 1.2
Subconcept: Social Interactions (SI)	
4.IC.SI.1	**Seek opportunities for local and global collaboration to facilitate communication and innovation.** *Computing influences many social institutions such as family, education, religion, and the economy. People can work in different places and at different times to collaborate and share ideas when they use technologies that reach across the globe. Computing provides the possibility for collaboration and sharing of ideas and allows the benefit of diverse perspectives. These social interactions affect how local and global groups interact with each other, and alternatively, these interactions can change the nature of groups. For example, a class can discuss ideas in the same class, school, or in another state or nation through interactive webinars.* Practice(s): Fostering an Inclusive Computing Culture: 1.1
Subconcept: Safety, Law, and Ethics (SLE)	
4.IC.SLE.1	**Use public domain or creative commons media, and refrain from copying or using material created by others without permission.** *Ethical complications arise from the opportunities provided by computing. The ease of sending and receiving copies of media~~,~~ , such as video, photos, and music, on the Internet~~, such as video, photos, and music,~~ creates the opportunity for unauthorized use, such as online piracy and disregard of copyrights. Students should consider the licenses ~~on~~ for the computational artifacts that they wish to use. For example, the license on a downloaded image or audio file may have restrictions that prohibit modification, require attribution, or prohibit use entirely.* Practice(s): Communicating About Computing: 7.3

Fifth Grade

By the end of fifth grade, students independently and collaboratively construct programs and ~~utilize~~ use algorithms to accomplish real world tasks. Students continue to decompose larger problems into smaller tasks. They also recognize the impacts of computing and computing devices and model how computing systems work. The computer science literate student understands that accurate use of terminology as well as the responsible use of technology is vital and continues to ~~place emphasis on~~recognize and attribute intellectual property rights appropriately.

Essential Concept: Computing Systems (CS)

Subconcept: Devices (D)	
5.CS.D.1	**Analyze and model how internal and external parts of computing devices communicate as a system.** *Computing devices often depend on other devices or components. For example, a robot depends on a physically attached light sensor to detect changes in brightness, whereas the light sensor depends on the robot for power. A smartphone can use wirelessly connected headphones to send audio information, and the headphones ~~are useless without~~require a music source.* *Practice(s): Communicating About Computing, Recognizing and Defining Computational Problems, Creating Computational Artifacts, Testing and Refining Computational Artifacts: 7.2, 3.2, 5.2, 6.3*
5.CS.D.2	**Computing devices affect humans in positive and negative ways.** *The use of computing devices ~~also~~ has potential consequences~~, such as in the areas of~~, especially with regard to privacy and security.* *Practice(s): Fostering an Inclusive Computing Culture, Communicating About Computing: 1.1, 7.2*
Subconcept: Hardware and Software (HS)	
5.CS.HS.1	**Model how information is transformed into binary digits to be stored or processed.** *Hardware and software communicate in binary digits commonly represented in 0s and 1s. For example, the roman numeral eight (8) can be converted to its binary equivalent (1 0 0 0).* *Practice(s): Communicating About Computing, Creating Computational Artifacts: 7.2, 5.2*

Commented [MOU13]: How are they creating artifacts here? Are they actually creating models?

5.CS.HS.2	**Provide examples of how hardware can accomplish different tasks depending on the software.** *In order for a person to accomplish tasks with a computer, both hardware and software are needed. At this stage, a model should only include the basic elements of a computer system, such as input, output, processor, sensors, and storage. Students could draw a model on paper or in a drawing program, program an animation to demonstrate it, or demonstrate it by acting ~~this~~ it out in some way.* *Practice(s): Communicating About Computing, Creating Computational Artifacts: 7.2, 5.3*
Subconcept: Troubleshooting (T)	
5.CS.T.1	**Independently apply potential solutions and solve simple hardware and software problems using common troubleshooting strategies.** *Although computing systems may vary, common troubleshooting strategies such as checking connections and power or swapping a working part in place of a potentially defective part can be used to restore functionality~~can be used on them, such as checking connections and power or swapping a working part in place of a potentially defective part~~. Restarting a computer (rebooting) is commonly effective because it resets the machine.~~Rebooting a machine is commonly effective because it resets the computer.~~* *Because computing devices are composed of an interconnected system of hardware and software, troubleshooting strategies may need to address both.* *Practice(s): Recognizing and Defining Computational Problems, Developing and Using Abstractions: 3.2, 4.1*

Commented [MOU14]: How is this developing or using abstraction?

Essential Concept: Networks and the Internet (NI)

Subconcept: Cybersecurity (C)	
5.NI.C.1	**Identify solutions to real-world cybersecurity problems ~~and how~~the protection of personal information ~~can be protected~~.** *Just as we protect our personal property on line, we also need to protect our devices and the information stored on them. Information can be protected using various security measures. These measures can be physical and/or digital. For example, discussion topics could be based on current events related to cybersecurity or topics that are applicable to students and the programs/devices they use.* *Practice(s): Communicating about Computing, Recognizing and Defining Computational Problems: 7.2, 3.1*
Subconcept: Network, Communication, and Organization (NCO)	

5.NI. NCO.1	**Analyze and evaluate the advantages and disadvantages of various network types.** *There are physical paths for communicating information, such as Ethernet cables, and wireless paths, such as Wi-Fi or cellular data. The choice of device and type of connection will affect the path information travels and the potential bandwidth (the capacity to transmit data or bits in a given timeframe).* *Practice(s): Developing and Using Abstractions, Collaborating Around Computing: 4.1, 2.4*

Commented [MOU15]: How is the developing and using abstraction?

Essential Concept: Data and Analysis (DA)

Subconcept: Collection, Visualization and Transformation (CVT)	
5.DA. CVT.1	**Independently select tools to collect, organize, manipulate, and present data visually through multiple representations to highlight relationships and support a claim.** *Tools are chosen based upon the type of measurement they use as well as the type of data people wish to observe. Organizing data can make interpreting and communicating it to others easier. Data points can be clustered by a number of commonalities. The same data could be manipulated and displayed in different formats to emphasize particular aspects or parts of the data set.* *Practice(s): Developing and Using Abstractions, Creating Computational Artifacts: 4.1, 5.1*
Subconcept: Storage (S)	
5.DA.S.1	**Identify and discuss different file extensions and how they are stored and retrieved on a computing device.** *Music, images, video, and text require different amounts of storage. Video will often require more storage than music or images alone because video combines both. For example, two pictures of the same object can require different amounts of storage based upon their resolution.* *Practice(s): Communicating About Computing: 7.2*
Subconcept: Inference and Models (IM)	
5.DA.IM. 1	**Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea.** *Explanation: People use data to highlight or propose cause-and-effect relationships and predict outcomes. Basing inferences or predictions on data does not guarantee their accuracy; the data must be relevant and of sufficient quantity.* *Practice(s): Communicating About Computing, Developing and Using Abstractions, Collaborate around Computing: 7.1, 4.3, 2.4*

Commented [MOU16]: Where is collaboration present in this standard?

Essential Concept: Algorithms and Programming (AP)

Subconcept: Algorithms (A)	
5.AP.A.1	**Compare, test, and refine multiple algorithms for the same task and determine which is the most appropriate.** Different algorithms can achieve the same result, though sometimes one algorithm might be most appropriate for a specific situation. Students should be able to look at different ways to solve the same task and decide which would be the best solution. For example, students could use a map and plan multiple algorithms to get from one point to another. They could look at routes suggested by mapping software and change the route to something that would be better, based on which route is shortest or fastest or would avoid a problem. Students might compare algorithms that describe how to get ready for school. Another example might be to write different algorithms to draw a regular polygon and determine which algorithm would be the easiest to modify or repurpose to draw a different polygon. Students test their algorithms to verify their effectiveness. *Practice(s):* Testing and Refining Computational Artifacts, Recognizing and Defining Computational Problems: 6.1, 6.3
Subconcept: Variables (V)	
5.AP.V.1	**Create programs that use variables to store and modify data, recognizing that the data type determines the values that can be stored and the operations that can be performed on the data.** *Variables are the vehicle through which computer programs store different types of data. At this level, understanding how to use variables is sufficient, without a fuller understanding of the technical aspects of variables (such as identifiers and memory locations). Data types vary by programming language, but many have types for numbers and text. Examples of operations associated with those types ~~are~~ include multiplying numbers and combining text. Some visual, block-based languages do not have explicitly declared types but still have certain operations that apply only to particular types of data in a program. Programs can imply either digital or paper-based ~~paper based~~ designs.* *Practice(s):* Creating Computational Artifacts: 5.2
Subconcept: Control (C)	
5.AP.C.1	**Create programs that include sequences, events, loops, and conditionals.** *Control structures specify the order (sequence) in which instructions are executed within a program and can be combined to support the creation of more complex programs. For example, if dialogue is not sequenced correctly when programming a simple animated story, the story will not make sense. If the commands to program a robot are not in the correct order, the robot will not complete the desired task~~task desired~~. Events allow portions of a program to run based on a specific action. For example, students could write a program to explain the water cycle and when a specific component is clicked (event), the program would show information about*

Commented [MOU17]: Do you really mean "appropriate" or do you mean "effective"?

	that part of the water cycle. Loops allow for the repetition of a sequence of code multiple times. For example, in a program that produces an animation about a famous historical character, students could use a loop to have the character walk across the screen as they introduce themselves. Conditionals allow for the execution of a portion of code in a program when a certain condition is true. For example, students could write a math game that asks multiplication fact questions and then uses a conditional to check whether or not the answer that was entered is correct. Practice(s): Creating Computational Artifacts: 5.2
Subconcept: Modularity (M)	
5.AP.M.1	**Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.** *Decomposition is the act of breaking down ~~tasks~~ a task into multiple simpler tasks. Decomposition also enables different people to work on different parts at the same time. For example, students could create an animation by separating a story into different scenes. For each scene, they would select a background, place characters, and program actions.* Practice(s): Recognizing and Defining Computational Problems: 3.2
5.AP.M.2	**Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.** *Programs can be broken down into smaller parts, which can be incorporated into new or existing programs. For example, students could modify prewritten code from a single-player game to create a two-player game with slightly different rules, remix and add another scene to an animated story, use code to make a ball bounce from another program in a new basketball game, or modify an image created by another student.* Practice(s): Creating Computational Artifacts: 5.3
Subconcept: Program Development (PD)	
5.AP.PD.1	**Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences.** *Explanation: Planning is an important part of the iterative process of program development. Students outline key features, time and resource constraints, and user expectations. Students should document the plan as, for example, a storyboard, flowchart, pseudocode, or story map.* Practice(s): Fostering an Inclusive Computing Culture, Creating Computational Artifacts: 1.1, 5.1

5.AP.PD.2	**Observe intellectual property rights and give appropriate attribution when creating or remixing programs.** *Explanation: Intellectual property rights can vary by country but copyright laws give the creator of a work a set of rights that prevents others from copying the work and using it in ways that the creator ~~they~~ may not like. Students should identify instances of remixing, when ideas are borrowed and iterated upon, and credit the original creator. Students should also consider common licenses that place limitations or restrictions on the use of computational artifacts, such as images and music downloaded from the Internet. At this stage, attribution should be written in the format required by the teacher and should always be included on any programs shared online.* Practice(s): Creating Computational Artifacts, Communicating About Computing: 5.2, 7.3
5.AP.PD.3	**Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.** *Explanation: As students develop programs they should continuously test those programs to see that they do what was expected and fix (debug)~~,~~ any errors. Students should also be able to successfully debug simple errors in programs created by others.* Practice(s): Testing and Refining Computational Artifacts: 6.1, 6.2
5.AP.PD.4	**Take on varying roles when collaborating with peers during the design, implementation, and review stages of program development.** *Explanation: Collaborative computing is the process of performing a computational task by working in pairs or on teams. Because it involves asking for the contributions and feedback of others, effective collaboration can lead to better outcomes than working independently. Students should take turns in different roles during program development, such as note taker, facilitator, program tester, or "driver" of the computer.* Practice(s): Collaborating Around Computing: 2.2
5.AP.PD.5	**Describe choices made during program development using code comments, presentations, and demonstrations.** *Explanation: People communicate about their code to help others understand and use their programs. Another purpose of communicating one's design choices is to show an understanding of one's work. These explanations could manifest themselves as in-line code comments for collaborators and assessors, or as part of a summative presentation, such as a code walk-through or coding journal.* Practice(s): Communicating About Computing: 7.2

Essential Concept: Impacts of Computing (IC)

Subconcept: Culture (C)

5.IC.C.1	**Discuss computing technologies that have changed the world, and express how those technologies influence, and are influenced by, cultural practices.** *Explanation: New computing technology is created and existing technologies are modified for many reasons, including ~~in order~~ to increase their benefits, decrease their risks, and meet societal needs. Students discuss topics that relate to the history of technology and the changes in the world due to technology. Topics could be based on current news content in areas~~,~~ such as robotics, wireless Internet, mobile computing devices, GPS systems, wearable computing, or how social media has influenced social and political changes.* Practice(s): Recognizing and Defining Computational Problems: 3.1
5.IC.C.2	**Brainstorm and design ways to improve the accessibility and usability of technology products for the diverse needs and wants of users.** *Explanation: The development and modification of computing technology are driven by people's needs and wants and can affect groups differently. Anticipating the needs and wants of diverse end users requires students to purposefully consider potential perspectives of users with different backgrounds, ability levels, points of view, and disabilities. For example, students may consider using both speech and text when they wish to convey information in a game. They may also wish to vary the types of programs they create, knowing that not everyone ~~share~~ shares their own tastes.* Practice(s): Fostering an Inclusive Computing Culture: 1.2
Subconcept: Social Interactions (SI)	
5.IC.SI.1	**Seek opportunities for local and global collaboration to facilitate communication and innovation.** *Computing influences many social institutions such as family, education, religion, and the economy. People can work in different places and at different times to collaborate and share ideas when they use technologies that reach across the globe. Computing provides the possibility for collaboration and sharing of ideas and allows the benefit of diverse perspectives. These social interactions affect how local and global groups interact with each other, and alternatively, these interactions can change the nature of groups. For example, a class can discuss ideas in the same class, school, or in another state or nation through interactive webinars.* Practice(s): Fostering an Inclusive Computing Culture: 1.1
Subconcept: Safety, Law, and Ethics (SLE)	
5.IC. SLE.1	**Use public domain or creative commons media, and refrain from copying or using material created by others without permission.** *Ethical complications arise from the opportunities provided by computing. The ease of sending and receiving copies of media, such as video, photos, and music, on the Internet~~, such as video, photos, and music~~, creates the opportunity for unauthorized use, such as online piracy and disregard of copyrights. Students should consider the licenses ~~on~~ for the computational artifacts that they wish to*

	use. For example, the license on a downloaded image or audio file may have restrictions that prohibit modification, require attribution, or prohibit use entirely. Practice(s): Communicating About Computing: 7.3

Sixth Grade

By the end of sixth grade, students will explore how devices process data and address potential problems. They will investigate the process of data transmission and the need for security concerns. Individually and in small groups, they will consider the reliability and validity of computational models used to process and represent data. They will identify possible solutions to programming challenges based on user needs. Successful students can ~~implement programming skills~~develop programs using parameters to meet a project's goal and timeline. Computer science literate students will be able to identify the advantages and disadvantages of computing technologies in everyday activities, including bias, accessibility, and privacy.

Essential Concept: Computing Systems (CS)

Subconcept: Devices (D)	
6.CS.D.1	**Compare how computing devices are designed based on an analysis of how users interact with devices.** *The study of human–computer interaction (HCI) can improve the design of devices, including both hardware and software. Teachers can guide students to consider usability through several lenses.* *Practice(s): Recognizing and Defining Computational Problems: 3.3*
Subconcept: Hardware and Software (HS)	
6.CS.HS.1	**Explain how hardware and software can be used to collect and exchange data.** *Collecting and exchanging data involves input, output, storage, and processing. For example, components for a mobile app could include an accelerometer, GPS, and speech recognition.* Practice(s): Creating Computational Artifacts: 5.1
Subconcept: Troubleshooting (T)	
6.CS.T.1	**Identify problems that can occur in computing devices and their ~~other~~ components within a system.** *Since a computing device may interact with interconnected devices within a system, problems may not be due to ~~the~~ the computer itself but to the components or ~~specific computing device itself but to~~ devices connected to it.* Practice(s): Testing and Refining Computational Artifacts: 6.2

Commented [MOU18]: Suggest: "Describe how user interaction testing can inform the design of computing tools and systems.

Essential Concept: Networks and the Internet (NI)

Subconcept: Cybersecurity (C)	
6.NI.C.1	**Identify multiple methods of encryption to secure the transmission of information.** *Explanation: Encryption can be as simple as letter substitution or as complicated as modern methods used to secure networks and the Internet. The students will identify different methods of encoding and decoding for encryptions used to hide or secure information.* *Practice(s):* Developing and Using Abstractions: 4.4
6.NI.C.2	**Identify different physical and digital security measures that protect electronic information.** *Explanation: Information that is stored online is vulnerable to unwanted access. Examples of physical security measures to protect data include keeping passwords hidden, locking doors, making backup copies on external storage devices, and erasing a storage device before it is reused. Examples of digital security measures include secure router admin passwords, firewalls that limit access to private networks, and the use of a protocol such as HTTPS to ensure secure data transmission.* Practice(s): Communicating About Computing: 7.2
Subconcept: Network, Communication, and Organization (NCO)	
6.NI. NCO.1	**Describe how protocols are used in transmitting data across networks and the Internet.** *Explanation: Protocols are rules that define how messages are sent between computers ~~are sent~~. They determine how quickly and securely information is transmitted across networks and the Internet, as well as how to check for and handle errors in transmission. Students should describe how data is sent using protocols to choose the fastest path, to deal with missing information, and to deliver sensitive data securely. The priority at this level is understanding the purpose of protocols and how they enable secure and errorless communication. Knowledge of the details of how specific protocols work is not expected.* Practice(s): Developing and Using Abstractions: 4.4

Essential Concept: Data and Analysis (DA)

Subconcept: Collection, Visualization and Transformation (CVT)	
6.DA. CVT.1	**Compare different computational tools used to collect data that is meaningful and useful.** *As students continue to explore ways organize and present data visually to support a claim, they will need to understand when and how to transform data for this purpose.* Practice(s): Testing and Refining Computational Artifacts: 6.3

Subconcept: Storage (S)	
6.DA.S.1	**Identify multiple encoding schemes used to represent data, including binary and ASCII.** *Students should explore the same data in multiple ways. For example, students could represent the same color using binary, RGB values, hex codes (low-level representations), ~~as well~~or as forms understandable by people, including words, symbols, and digital displays of the color (high-level representations).* Practice(s): Developing and Using Abstractions: 4.0
Subconcept: Inference and Models (IM)	
6.DA. IM.1	**Evaluate computational models based on reliability and validity.** *A model may be a programmed simulation of events or a representation of how various data is related. ~~In order~~T ~~t~~o refine a model, students need to consider which data points are relevant, how data points relate to each other, and if the data is accurate.* Practice(s): Creating Computational Artifacts, Developing and Using Abstractions: 5.3, 4.4

Essential Concept: Algorithms and Programming (AP)

Subconcept: Algorithms (A)	
6.AP.A.1	**Identify planning strategies such as flowcharts or pseudocode, to help express ~~address~~ problems as algorithms.** *Students should recognize planning strategies to organize and sequence an algorithm that addresses a problem, even though they may not actually program the solutions. For example, students might express an algorithm that produces a recommendation for purchasing sneakers based on inputs such as size, colors, brand, comfort, and cost.* *Practice(s): Developing and Using Abstractions: 4.4, 4.1*
Subconcept: Variables (V)	
6.AP.V.1	**Identify variables that represent different data types and perform operations on their values.** *A variable is like a container with a name, in which the contents may change, but the name (identifier) does not. When planning and developing programs, students should decide when and how to declare and name new variables. Students should use naming conventions to improve program readability. Examples of possible operations include adding points to the score, combining user input with words to make a sentence, changing the size of a picture, or adding a name to a list of people.* *Practice(s): Creating Computational Artifacts: 5.1, 5.2*

Subconcept: Control (C)	
6.AP.C.1	**Design programs that combine control structures, including nested loops and compound conditionals.** *Control structures can be combined in many ways. Nested loops are loops placed within loops. Compound conditionals combine two or more conditions in a logical relationship (e.g., using AND, OR, and NOT), and nesting conditionals within one another allows the result of one conditional to lead to another. For example, when programming an interactive story, students could use a compound conditional within a loop to unlock a door only if a character has a key AND is touching the door.* *Practice(s): Creating Computational Artifacts: 5.1, 5.2*
Subconcept: Modularity (M)	
6.AP.M.1	**Decompose problems into subparts problems to facilitate the design, implementation, and review of programs.** *Decompose problems into parts to facilitate the design, implementation, and review of programs.* *Practice(s): Recognizing and Defining Computational Problems: 3.2*
6.AP.M.2	**Use procedures to organize code and make it easier to reuse.** *Students should compare procedures and/or functions that are used multiple times within a program to repeat groups of instructions. These procedures can be generalized by defining parameters that create different outputs for a wide range of inputs. For example, a procedure to draw a circle involves many instructions, but all of them can be invoked with one instruction, such as "drawCircle." By adding a radius parameter, the user can easily draw circles of different sizes.* *Practice(s): Developing and Using Abstractions: 4.1, 4.3*
Subconcept: Program Development (PD)	
6.AP.PD.1	**Seek and incorporate feedback from team members and users to refine a solution that meets user needs.** *Development teams that employ user-centered design create solutions (e.g., programs and devices) that can have a large societal impact, such as an app that allows people with speech difficulties to translate hard-to-understand pronunciation into understandable language. Students should begin to seek diverse perspectives throughout the design process to improve their computational artifacts. Considerations of the end-user may include usability, accessibility, age-appropriate content, respectful language, user perspective, pronoun use, color contrast, and ease of use.* *Practice(s): Collaborating Around Computing, Fostering an Inclusive Computing Culture: 2.3, 1.1*
6.AP.PD.2	**Incorporate existing code into programs and give attribution.** *Building on the work of others enables students to produce more interesting and powerful creations. Students should use portions of code in their own programs and websites. For example, when creating a side-scrolling game, students may incorporate portions*

	of code that create a realistic jump movement from another person's game~~, and~~. T ~~t~~hey may also import Creative Commons-licensed images to use in the background. Students should give attribution to the original creators to acknowledge their contributions. *Practice(s): Developing and Using Abstractions, Creating Computational Artifacts, Communicating About Computing: 4.2, 5.2, 7.3*
6.AP.PD.3	**Test programs using a range of inputs and identify expected outputs.** *At this level, testing should become a deliberate process that is more iterative, systematic, and proactive than at lower levels.* *Practice(s): Testing and Refining Computational Artifacts: 6.1*
6.AP.PD.4	**Adhere to a project timeline with specific tasks while collaboratively developing computational artifacts.** *Collaboration is a common and crucial practice in program~~ming~~ development. Often, many individuals and groups work on the interdependent parts of a project together. Students should assume pre-defined roles within their teams and manage the project workflow using structured timelines.* *Practice(s): Collaborating Around Computing: 2.2*
6.AP.PD.5	**Document programs to make them easier to follow, test, and debug.** *Documentation allows creators and others to more easily use and understand a program. Students should provide documentation for end users that explain~~s~~ their artifacts and how they function. For example, students could provide a project overview and clear user instructions. They should also incorporate comments ~~in their product~~into their programs and communicate their process throughout the design, development, and user experience phases.* *Practice(s): Communicating About Computing: 7.2*

Essential Concept: Impacts of Computing (IC)

Subconcept: Culture (C)	
6.IC.C.1	**Identify some of the tradeoffs associated with computing technologies that can affect people's everyday activities and career options.** Advancements in computer technology are neither wholly positive nor negative. However, the ways that people use computing technologies have tradeoffs. Students should consider current events related to broad ideas, including privacy, communication, and automation. For example, driverless cars can increase convenience and reduce accidents, but they are also susceptible to hacking. The emerging industry will reduce the number of taxi and shared-ride drivers but will create more software engineering and cybersecurity jobs. Practice(s): Communicating About Computing: 7.2

6.IC.C.2	**Identify ~~issues of~~how bias and accessibility issues can impact the functionality ~~in the design~~ of existing technologies.** Students should identify, with teacher's guidance, how various technological tools have different levels of usability. For example, facial recognition software that works better for certain skin tones was likely developed with a homogeneous testing group and could be improved by sampling a more diverse population. ~~Identifying~~ Ways of improving the accessibility ~~to~~ of technological tools can include allowing a user to change font sizes and colors. This will ~~not only~~ make an interface usable for people with low vision and ~~but also~~ benefit~~s~~ users in ~~various~~ situations~~,~~ such as in bright daylight or a dark room. Practice(s): Fostering an Inclusive Computing Culture: 1.2
Subconcept: Social Interactions (SI)	
6.IC.SI.1	**Identify the advantages of using digital technologies to collaborate with others while creating a computational product ~~by collaborating with others using digital technologies~~.** Different digital technologies can be used to gather services, ideas, or content from a large group of people, especially from the online community. It can be done at the local level (e.g., classroom or school) or global level (e.g., age-appropriate online communities). For example, a group of students could combine animations to produce a digital community creation. They could also solicit feedback from many people though use of online communities and electronic surveys. Practice(s): Collaborating Around Computing, Creating Computational Artifacts: 2.4, 5.2
Subconcept: Safety, Law, and Ethics (SLE)	
6.IC. SLE.1	**Describe how some digital information can be public or can be kept private and secure.** Sharing information online can help establish, maintain, and strengthen connections between people. Students should consider current events related to broad ideas, including privacy, communication, and automation. Practice(s): Communicating About Computing: 7.2

Seventh Grade

By the end of seventh grade, students can evaluate how devices process data and address potential problems through project development. They will investigate the need for security measures and protocols for data transmission. Successful students will be able to integrate reliable and valid computational models to process data that is meaningful and useful. They can evaluate possible solutions to programming challenges based on the user's needs. Students can compare and contrast possible solutions utilizing computing technologies to solve everyday challenges, taking into consideration bias, accessibility, and privacy. Computer science literate students will ~~implement programming skills~~develop programs using parameters to meet a project's goal and timeline.

Essential Concept: Computing Systems (CS)

Subconcept: Devices (D)	
7.CS.D.1	**Identify some advantages, disadvantages, and consequences with the design of computer devices based on an analysis of how users interact with devices.** *The study of human–computer interaction (HCI) can improve the design of devices, including both hardware and software. Teachers can guide students to consider usability through several lenses, including accessibility, ergonomics, and learnability. For example, assistive devices provide capabilities such as scanning written information and converting it to speech.* *Practice(s): Recognizing and Defining Computational Problems: 3.3*
Subconcept: Hardware and Software (HS)	
7.CS.HS.1	**Design projects that combine hardware and software to collect and exchange data.** *Collecting and exchanging data involves input, output, storage, and processing. When possible, students should select the hardware and software components for their project designs by considering factors such as functionality, cost, size, speed, accessibility, and aesthetics. For example, components for a mobile app could include an accelerometer, GPS, and speech recognition.* Practice(s): Creating Computational Artifacts: 5.1
Subconcept: Troubleshooting (T)	

Commented [MOU19]: This standard is very confusing and it is not clear what the primary learning objective is. Is the intent to have them analyze users interactions with devices to inform their understanding of the consequences of design decisions or to have them propose specific advantages, disadvantages or consequences of design decisions on a diversity of users?

7.CS.T.1	**Evaluate solutions to fix problems with computing devices and their ~~other~~ components within a system.** *Since a computing device may interact with interconnected devices within a system, problems may not be due to the computer itself but to the components or ~~the specific computing device itself but to~~ devices connected to it. Examples of troubleshooting strategies include following a troubleshooting flow diagram, making changes to software to see if hardware will work, checking connections and settings, and swapping in working components.* Practice(s): Testing and Refining Computational Artifacts: 6.2

Essential Concept: Networks and the Internet (NI)

Subconcept: Cybersecurity (C)	
7.NI.C.1	**Evaluate multiple methods of encryption to model the secure transmission of information.** Encryption can be as simple as letter substitution or as complicated as modern methods used to secure networks and the Internet. The students will examine the different levels of complexity used to hide or secure information. For example, students explore different methods of securing messages using methods such as Caesar cyphers or steganography (i.e., hiding messages inside a picture or other data). *Practice(s):* Developing and Using Abstractions: 4.4
7.NI.C.2	**Explain how physical and digital security measures protect electronic information.** Information that is stored online is vulnerable to unwanted access. Examples of physical security measures to protect data include keeping passwords hidden, locking doors, making backup copies on external storage devices, and erasing a storage device before it is reused. Examples of digital security measures include secure router admin passwords, firewalls that limit access to private networks, and the use of a protocol such as HTTPS to ensure secure data transmission. Practice(s): Communicating About Computing: 7.2
Subconcept: Network, Communication, and Organization (NCO)	
7.NI. NCO.1	**Compare and contrast models to illustrate the role of protocols in transmitting data across networks and the Internet.** Protocols are rules that define how messages are sent between computers ~~are sent~~. They determine how quickly and securely information is transmitted across networks and the Internet, as well as how to check for and handle errors in transmission. Students should examine how data is sent using protocols to choose the fastest path, to deal with missing information, and to deliver sensitive data securely. The priority at this level is understanding the purpose of protocols and how they enable secure and errorless communication. Knowledge of the details of how specific protocols work is not expected.

Commented [MOU20]: The explanations in this section are not italicized as they are in other sections.

	Practice(s): Developing and Using Abstractions: 4.4

Essential Concept: Data and Analysis (DA)

Subconcept: Collection, Visualization and Transformation (CVT)	
7.DA.CVT.1	**Collect data using computational tools to create models that are meaningful and useful.** *As students continue to build on their ability to organize and present data visually to support a claim, they will need to understand when and how to transform data for this purpose.* Practice(s): Testing and Refining Computational Artifacts: 6.3
Subconcept: Storage (S)	
7.DA.S.1	~~Utilize~~ Use **multiple encoding schemes to represent data, including binary and ASCII.** *Students should represent the same data in multiple ways. For example, students could represent the same color using binary, RGB values, hex codes (low-level representations), as well as forms understandable by people, including words, symbols, and digital displays of the color (high-level representations).* Practice(s): Developing and Using Abstractions: 4.0
Subconcept: Inference and Models (IM)	
7.DA.IM.1	**Design computational models based on reliability and the validity of the data they generate.** *A model may be a programmed simulation of events or a representation of how various data are related. ~~In order t~~To refine a model, students need to consider which data points are relevant, how data points relate to each other, and if the data are accurate. For example, students may make a prediction about how far a ball will travel based on a table of data related to the height and angle of a track.* Practice(s): Creating Computational Artifacts, Developing and Using Abstractions: 5.3, 4.4

Essential Concept: Algorithms and Programming (AP)

Subconcept: Algorithms (A)

7.AP.A.1	**Use planning strategies, such as flowcharts or pseudocode, to develop algorithms to address complex problems ~~as algorithms~~.** *Complex problems are problems that would be difficult for students to solve computationally. Students should use pseudocode and/or flowcharts to organize and sequence an algorithm that addresses a complex problem, even though they may not actually program the solutions. For example, students might follow an algorithm that produces a recommendation for purchasing sneakers based on inputs such as size, colors, brand, comfort, and cost.* *Practice(s): Developing and Using Abstractions: 4.4, 4.1*
Subconcept: Variables (V)	
7.AP.V.1	**Compare and contrast variables that represent different data types and perform operations on their values.** *A variable is like a container with a name, in which the contents may change, but the name (identifier) does not. When planning and developing programs, students should decide when and how to declare and name new variables. Students should use naming conventions to improve program readability. Examples of possible operations include adding points to the score, combining user input with words to make a sentence, changing the size of a picture, or adding a name to a list of people.* *Practice(s): Creating Computational Artifacts: 5.1, 5.2*
Subconcept: Control (C)	
7.AP.C.1	**Design and develop programs that combine control structures, including nested loops and compound conditionals.** *Control structures can be combined in many ways. Nested loops are loops placed within loops. Compound conditionals combine two or more conditions in a logical relationship (e.g., using AND, OR, and NOT), and nesting conditionals within one another allows the result of one conditional to lead to another. For example, when programming an interactive story, students could use a compound conditional within a loop to unlock a door only if a character has a key AND is touching the door.* *Practice(s): Creating Computational Artifacts: 5.1, 5.2*
Subconcept: Modularity (M)	
7.AP.M.1	**Decompose problems into ~~parts~~ subproblems to facilitate the design, implementation, and review of programs.** *Decompose problems into parts to facilitate the design, implementation, and review of programs.* *Practice(s): Recognizing and Defining Computational Problems: 3.2*
7.AP.M.2	**Use procedures with parameters to organize code and make it easier to reuse.** *Students should use procedures and/or functions that are used multiple times within a program to repeat groups of instructions. These procedures can be generalized by defining parameters that create different outputs for a wide range of inputs. For example, a*

	procedure to draw a circle involves many instructions, but all of them can be invoked with one instruction, such as "drawCircle." By adding a radius parameter, the user can easily draw circles of different sizes. *Practice(s): Developing and Using Abstractions: 4.1, 4.3*
Subconcept: Program Development (PD)	
7.AP.PD.1	**Seek and incorporate feedback from team members and users to refine a solution that meets user needs.** *Development teams that employ user-centered design create solutions (e.g., programs and devices) that can have a large societal impact, such as an app that allows people with speech difficulties to translate hard-to-understand pronunciation into understandable language. Students should begin to seek diverse perspectives throughout the design process to improve their computational artifacts. Considerations of the end-user may include usability, accessibility, age-appropriate content, respectful language, user perspective, pronoun use, color contrast, and ease of use.* *Practice(s): Collaborating Around Computing, Fostering an Inclusive Computing Culture: 2.3, 1.1*
7.AP.PD.2	**Incorporate existing code and media into programs and give attribution.** *Building on the work of others enables students to produce more interesting and powerful creations. Students should use portions of code and/or digital media in their own programs and websites. For example, when creating a side-scrolling game, students may incorporate portions of code that create a realistic jump movement from another person's game~~, and t~~. T*hey may also import *Creative Commons-licensed images to use in the background. Students should give attribution to the original creators to acknowledge their contributions.* *Practice(s): Developing and Using Abstractions, Creating Computational Artifacts, Communicating About Computing: 4.2, 5.2, 7.3*
7.AP.PD.3	**Systematically test and refine programs using a range of possible inputs.** *At this level, testing should become a deliberate process that is more iterative, systematic, and proactive than at lower levels. Students should begin to test programs by considering potential errors, such as what will happen if a user enters invalid input (e.g., negative numbers and 0 instead of positive numbers).* *Practice(s): Testing and Refining Computational Artifacts: 6.1*
7.AP.PD.4	**Distribute and execute tasks while maintaining a project timeline when collaboratively developing computational artifacts.** *Collaboration is a common and crucial practice in program~~ming~~ development. Often, many individuals and groups work on the interdependent parts of a project together. Students should assume pre-defined roles within their teams and manage the project workflow using structured timelines. With teacher guidance, they will begin to create collective goals, expectations, and equitable workloads. For example, students may divide the design stage of a game into planning the storyboard, flowchart, and different parts of the game mechanics. They can then distribute tasks and roles among members of the team and assign deadlines.* *Practice(s): Collaborating Around Computing: 2.2*

Commented [MOU21]: Suggest also adding *Practice(s): Creating Computational Artifacts: 5.1, 5.2*

7.AP.PD.5	**Document programs ~~in order~~ to make them easier to follow, test, and debug.** *Documentation allows creators and others to more easily use and understand a program. Students should provide documentation for end users that explains their artifacts and how they function. For example, students could provide a project overview and clear user instructions. They should also incorporate comments into their ~~product~~ programs and communicate their process throughout the design, development, and user experience phases.* *Practice(s): Communicating About Computing: 7.2*

Essential Concept: Impacts of Computing (IC)

Commented [MOU22]: The explanations are not italicized in this section.

Subconcept: Culture (C)	
7.IC.C.1	**Explain how some of the tradeoffs associated with computing technologies can affect people's everyday activities and career options.** Advancements in computer technology are neither wholly positive nor negative. However, the ways that people use computing technologies have tradeoffs. Students should consider current events related to broad ideas, including privacy, communication, and automation. For example, driverless cars can increase convenience and reduce accidents, but they are also susceptible to hacking. The emerging industry will reduce the number of taxi and shared-ride drivers, but will create more software engineering and cybersecurity jobs. Practice(s): Communicating About Computing: 7.2
7.IC.C.2	**Discuss how bias and accessibility issues can impact the functionality of existing technologies.~~issues of bias and accessibility in the design of existing technologies.~~** Students should discuss the usability of various technology tools (e.g., apps, games, and devices) with the teacher's guidance. For example, facial recognition software that works better for certain skin tones was likely developed with a homogeneous testing group and could be improved by sampling a more diverse population. Ways of improving the accessibility of technological tools can include allowing a user to change font sizes and colors. This will make an interface usable for people with low vision and benefit users in situations such as in bright daylight or a dark room. ~~When discussing accessibility, students may notice that allowing a user to change font sizes and colors will not only make an interface usable for people with low vision but also benefits users in various situations, such as in bright daylight or a dark room.~~ Practice(s): Fostering an Inclusive Computing Culture: 1.2
Subconcept: Social Interactions (SI)	

7.IC.SI.1	**Describe the process for creating a computational product by collaborating with others using digital technologies.** Crowdsourcing can be used as a platform to gather services, ideas, or content from a large group of people, especially from the online community. It can be done at the local level (e.g., classroom or school) or global level (e.g., age-appropriate online communities). For example, a group of students could combine animations to produce a digital community creation. They could also solicit feedback from many people though use of online communities and electronic surveys. Practice(s): Collaborating Around Computing, Creating Computational Artifacts: 2.4, 5.2
Subconcept: Safety, Law, and Ethics (SLE)	
7.IC. SLE.1	**Identify the benefits and risks associated with sharing information digitally.** Sharing information online can help establish, maintain, and strengthen connections between people. For example, it allows artists and designers to display their talents and reach a broad audience. However, security attacks often start with personal information that is publicly available online. Social engineering is based on tricking people into revealing sensitive information and can be thwarted by being wary of attacks, such as phishing and spoofing. Practice(s): Communicating About Computing: 7.2

Eighth Grade

By the end of eighth grade, students will apply programming skills to process data and address problems using computing devices. They will implement security measures and protocols for data transmission to address vulnerabilities. Successful students can integrate possible solutions to programming challenges based on the user's needs. They will achieve this by ~~implementing programming skills~~creating programs using parameters to meet a project's goal and timeline. Students will be able to ~~utilize~~ use computing technologies and develop possible solutions to solve everyday challenges, taking into consideration bias, accessibility, and privacy. Computer science literate students will be able to collect and represent reliable and valid computational models.

Essential Concept: Computing Systems (CS)

Subconcept: Devices (D)	
8.CS.D.1	**Improve the design of computing devices based on an analysis of how users interact with devices, including identifying that some advantages may have disadvantages and unintended consequences.** *The study of human–computer interaction (HCI) can improve the design of devices, including both hardware and software. Students should make recommendations for existing devices (e.g., a laptop, phone, or tablet) or design their own components or interface (e.g., create their own controllers). Teachers can guide students to consider usability through several lenses, including accessibility, ergonomics, and learnability. For example, assistive devices provide capabilities such as scanning written information and converting it to speech.* *Practice(s): Recognizing and Defining Computational Problems: 3.3*
Subconcept: Hardware and Software (HS)	
8.CS.HS.1	**Design and evaluate projects that combine hardware and software components to collect and exchange data.** *Collecting and exchanging data involves input, output, storage, and processing. When possible, students should select the hardware and software components for their project designs by considering factors such as functionality, cost, size, speed, accessibility, and aesthetics. For example, components for a mobile app could ~~includ~~include an~~e~~ accelerometer, GPS, and speech recognition. The choice of a device that connects wirelessly through a Bluetooth connection versus a physical USB connection involves a tradeoff between mobility and the need for an additional power source for the wireless device.* Practice(s): Creating Computational Artifacts: 5.1
Subconcept: Troubleshooting (T)	

8.CS.T.1	**Systematically identify and develop solutions to fix problems with computing devices and their components.** *Since a computing device may interact with interconnected devices within a system, problems may not be due to the ~~specific computing device~~computer itself but to the components or devices connected to it. Just as pilots use checklists to troubleshoot problems with aircraft systems, students should use a similar, structured process to troubleshoot problems with computing systems and ensure that potential solutions are not overlooked. Examples of troubleshooting strategies include following a troubleshooting flow diagram, making changes to software to see if hardware will work, checking connections and settings, and swapping in working components.* Practice(s): Testing and Refining Computational Artifacts: 6.2

Essential Concept: Networks and the Internet (NI)

Subconcept: Cybersecurity (C)	
8.NI.C.1	**Apply multiple methods of encryption to model the secure transmission of information.** *Explanation:* Encryption can be as simple as letter substitution or as complicated as modern methods used to secure networks and the Internet. Students should encode and decode messages using a variety of encryption methods, and they should understand the different levels of complexity used to hide or secure information. For example, students could secure messages using methods such as Caesar cyphers or steganography (i.e., hiding messages inside a picture or other data). They can also model more complicated methods, such as public key encryption, through unplugged activities. *Practice(s):* Developing and Using Abstractions: 4.4
8.NI.C.2	**Evaluate how various physical and digital security measures protect electronic information and how a lack of such measures could lead to vulnerabilities.** Information that is stored online is vulnerable to unwanted access. Examples of physical security measures to protect data include keeping passwords hidden, locking doors, making backup copies on external storage devices, and erasing a storage device before it is reused. Examples of digital security measures include secure router admin passwords, firewalls that limit access to private networks, and the use of a protocol such as HTTPS to ensure secure data transmission. Examples of vulnerabilities include password strength, awareness of how data is ~~utilized~~used, as well as threats to personal and professional data. Practice(s): Communicating About Computing: 7.2
Subconcept: Network, Communication, and Organization (NCO)	
8.NI.	**Develop models to illustrate the role of protocols in transmitting data across networks and the Internet.**

NCO.1	Protocols are rules that define how messages are sent between computers ~~are sent~~. They determine how quickly and securely information is transmitted across networks and the Internet, as well as how to check for and handle errors in transmission. Students should model how data is sent using protocols to choose the fastest path, to deal with missing information, and to deliver sensitive data securely. For example, students could devise a plan for resending lost information or for interpreting a picture that has missing pieces. The priority at this level is understanding the purpose of protocols and how they enable secure and errorless communication. Knowledge of the details of how specific protocols work is not expected. Practice(s): Developing and Using Abstractions: 4.4

Essential Concept: Data and Analysis (DA)

Subconcept: Collection, Visualization and Transformation (CVT)	
8.DA. CVT.1	**Collect data using computational tools and transform the data to make it more meaningful and useful.** *As students continue to build on their ability to organize and present data visually to support a claim, they will need to understand when and how to transform data for this purpose. Students should transform data to remove errors, highlight or expose relationships, and/or make it easier for computers to process. ~~The~~ Data cleaning ~~of data~~ is an important transformation for ensuring consistent format and reducing noise and errors (e.g., removing irrelevant responses in a survey). An example of a transformation that highlights a relationship is representing males and females as percentages of a whole instead of as individual counts.* Practice(s): Testing and Refining Computational Artifacts: 6.3
Subconcept: Storage (S)	
8.DA.S.1	**Represent data using multiple encoding schemes ~~through modeling~~, including binary and ASCII.** *Data representations occur at multiple levels of abstraction, from the physical storage of bits to the arrangement of information into organized formats (e.g., tables). Students should represent the same data in multiple ways. For example, students could represent the same color using binary, RGB values, hex codes (low-level representations), as well as forms understandable by people, including words, symbols, and digital displays of the color (high-level representations).* Practice(s): Developing and Using Abstractions: 4.0
Subconcept: Inference and Models (IM)	
8.DA.IM.	**Design and evaluate computational models based on the reliability and validity of the data they generate.**

1	*A model may be a programmed simulation of events or a representation of how various data is related. ~~In order to~~To refine a model, students need to consider which data points are relevant, how data points relate to each other, and if the data is accurate. For example, students may make a prediction about how far a ball will travel based on a table of data related to the height and angle of a track. The students could then test and refine their model by comparing predicted versus actual results and considering whether other factors are relevant (e.g., size and mass of the ball). Additionally, students could refine game mechanics based on test ~~outcomes in order~~ to make the game more balanced or fair.* Practice(s): Creating Computational Artifacts, Developing and Using Abstractions: 5.3, 4.4

Essential Concept: Algorithms and Programming (AP)

Subconcept: Algorithms (A)	
8.AP.A.1	**Develop planning strategies, such as flowcharts or pseudocode, to develop algorithms to address complex problems~~to address complex problems as algorithms.~~** *Complex problems are problems that would be difficult for students to solve computationally. Students should use pseudocode and/or flowcharts to organize and sequence an algorithm that addresses a complex problem, even though they may not actually program the solutions. For example, students might express an algorithm that produces a recommendation for purchasing sneakers based on inputs such as size, colors, brand, comfort, and cost. Testing the algorithm with a wide range of inputs and users allows students to refine their recommen~~datio~~ed~~n~~ algorithm and to identify other inputs they may have initially excluded.* *Practice(s): Developing and Using Abstractions: 4.4, 4.1*
Subconcept: Variables (V)	
8.AP.V.1	**Create clearly named variables that represent different data types and perform operations on their values.** *A variable is like a container with a name, in which the contents may change, but the name (identifier) does not. When planning and developing programs, students should decide when and how to declare and name new variables. Students should use naming conventions to improve program readability. Examples of operations include adding points to the score, combining user input with words to make a sentence, changing the size of a picture, or adding a name to a list of people.* *Practice(s): Creating Computational Artifacts: 5.1, 5.2*
Subconcept: Control (C)	

8.AP.C.1	**Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.** *Control structures can be combined in many ways. Nested loops are loops placed within loops. Compound conditionals combine two or more conditions in a logical relationship (e.g., using AND, OR, and NOT), and nesting conditionals within one another allows the result of one conditional to lead to another. For example, when programming an interactive story, students could use a compound conditional within a loop to unlock a door only if a character has a key AND is touching the door.* *Practice(s): Creating Computational Artifacts: 5.1, 5.2*
Subconcept: Modularity (M)	
8.AP.M.1	**Decompose problems into ~~parts~~ subproblems to facilitate the design, implementation, and review of programs.** *Programs use procedures to organize code, hide implementation details, and make code easier to reuse. Procedures can be repurposed in new programs.* *Practice(s): Recognizing and Defining Computational Problems: 3.2*
8.AP.M.2	**Create procedures with parameters to organize code and make it easier to reuse.** *Students should create procedures and/or functions that are used multiple times within a program to repeat groups of instructions. These procedures can be generalized by defining parameters that create different outputs for a wide range of inputs. For example, a procedure to draw a circle involves many instructions, but all of them can be invoked with one instruction, such as "drawCircle." By adding a radius parameter, the user can easily draw circles of different sizes.* *Practice(s): Developing and Using Abstractions: 4.1, 4.3*
Subconcept: Program Development (PD)	
8.AP.PD.1	**Seek and incorporate feedback from team members and users to refine a solution that meets user needs.** *Development teams that employ user-centered design create solutions (e.g., programs and devices) that can have a large societal impact, such as an app that allows people with speech difficulties to translate hard-to-understand pronunciation into understandable language. Students should begin to seek diverse perspectives throughout the design process to improve their computational artifacts. Considerations of the end-user may include usability, accessibility, age-appropriate content, respectful language, user perspective, pronoun use, color contrast, and ease of use.* *Practice(s): Collaborating Around Computing, Fostering an Inclusive Computing Culture: 2.3, 1.1*
8.AP.PD.2	**Incorporate existing code, media, and libraries into original programs, and give attribution.** *Building on the work of others enables students to produce more interesting and powerful creations. Students should use portions of code, algorithms, and/or digital media in their own programs and websites. At this level, they may also import libraries and connect to web application program interfaces (APIs). For example, when creating a side-scrolling game, students may incorporate*

Commented [MOU23]: The explanation contains no reference to decomposition.

	portions of code that create a realistic jump movement from another person's game. ~~, and~~ T~~t~~hey may also import Creative Commons-licensed images to use in the background. Students should give attribution to the original creators to acknowledge their contributions. *Practice(s): Developing and Using Abstractions, Creating Computational Artifacts, Communicating About Computing: 4.2, 5.2, 7.3*
8.AP.PD.3	**Systematically test and refine programs using a range of possible inputs.** *At this level, testing should become a deliberate process that is more iterative, systematic, and proactive than at lower levels. Students should begin to test programs by considering potential errors, such as what will happen if a user enters invalid input (e.g., negative numbers and 0 instead of positive numbers).* *Practice(s): Testing and Refining Computational Artifacts: 6.1*
8.AP.PD.4	**Distribute and execute tasks while maintaining a project timeline when collaboratively developing computational artifacts.** *Collaboration is a common and crucial practice in program~~ming~~ development. Often, many individuals and groups work on the interdependent parts of a project together. Students should assume pre-defined roles within their teams and manage the project workflow using structured timelines. With teacher guidance, they will begin to create collective goals, expectations, and equitable workloads. For example, students may divide the design stage of a game into planning the storyboard, flowchart, and different parts of the game mechanics. They can then distribute tasks and roles among members of the team and assign deadlines.* *Practice(s): Collaborating Around Computing: 2.2*
8.AP.PD.5	**Document programs ~~in order~~ to make them easier to follow, test, and debug.** *Documentation allows creators and others to more easily use and understand a program. Students should provide documentation for end users that explains their artifacts and how they function. For example, students could provide a project overview and clear user instructions. They should also incorporate comments in~~to~~ their ~~product~~ programs and communicate their process throughout the design, development, and user experience phases.* *Practice(s): Communicating About Computing: 7.2*

Essential Concept: Impacts of Computing (IC)

Subconcept: Culture (C)	
8.IC.C.1	**Compare and contrast tradeoffs associated with computing technologies that affect people's everyday activities and career options.** Advancements in computer technology are neither wholly positive nor negative. However, the ways that people use computing technologies have tradeoffs. Students should consider current events related to broad ideas, including privacy, communication, and

	automation. For example, driverless cars can increase convenience and reduce accidents, but they are also susceptible to hacking. The emerging industry will reduce the number of taxi and shared-ride drivers, but will create more software engineering and cybersecurity jobs. Practice(s): Communicating About Computing: 7.2
8.IC.C.2	**Develop a solution to address an issue of bias or accessibility in the design of existing technologies.** Students should test and discuss the usability of various technology tools (e.g., apps, games, and devices) with the teacher's guidance. For example, facial recognition software that works better for certain skin tones was likely developed with a homogeneous testing group and could be improved by sampling a more diverse population. When discussing accessibility, students may notice that allowing a user to change font sizes and colors will not only make an interface usable for people with low vision, but also benefits users in various situations, such as in bright daylight or a dark room. Practice(s): Fostering an Inclusive Computing Culture: 1.2
Subconcept: Social Interactions (SI)	
8.IC.SI.1	**Collaborate with contributors by ~~utilizing~~ using digital technologies when creating a computational product.** Crowdsourcing can be used as a platform to gather services, ideas, or content from a large group of people, especially from the online community. It can be done at the local level (e.g., classroom or school) or global level (e.g., age-appropriate online communities). For example, a group of students could combine animations to produce a digital community creation. They could also solicit feedback from many people though use of online communities and electronic surveys. Practice(s): Collaborating Around Computing, Creating Computational Artifacts: 2.4, 5.2
Subconcept: Safety, Law, and Ethics (SLE)	
8.IC. SLE.1	**Evaluate the benefits and risks associated with sharing information digitally.** Sharing information online can help establish, maintain, and strengthen connections between people. For example, it allows artists and designers to display their talents and reach a broad audience. However, security attacks often start with personal information that is publicly available online. Social engineering is based on tricking people into revealing sensitive information and can be thwarted by being wary of attacks, such as phishing and spoofing. Practice(s): Communicating About Computing: 7.2

High School

By the end of High School, students refine their skills in differentiating problems or subproblems that are best solved by computing systems or digital tools and those that are best solved by humans. Students further develop their computational thinking and problem-solving skills ~~which will facilitate~~to support the selection and appropriate use of technology. They can also evaluate security measures to protect sensitive data. Students will ~~utilize~~ use data visualizations to represent and communicate real-world phenomena. Successful students consider the scalability and reliability of networks~~, including~~ and data storage~~,~~ and ~~utilize~~ collaboratively use algorithms to develop computational solutions to solve real-world ~~challenges collaboratively;~~ ~~they~~ They implement computational solutions across disciplines, taking into consideration factors such as innovation, privacy, bias, and equity~~,~~. They also consider personal, ethical, social, economic, and cultural practices. Computer science literate students will be able to explore how abstractions are integrated in computing systems within software and hardware. They build on K–8 experiences and learn more technical and sophisticated applications.

Essential Concept: Computing Systems (CS)

Subconcept: Devices (D)	
HS.CS.D.1	**Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.** *Computing devices are often integrated with other systems, including biological, mechanical, and social systems. A medical device can be embedded inside a person to monitor and regulate his or her health, a hearing aid (a type of assistive device) can filter out certain frequencies and magnify others, a monitoring device installed in a motor vehicle can track a person's driving patterns and habits, and a facial recognition device can be integrated into a security system to identify a person. The usability, dependability, security, and accessibility of these devices, and the systems with which they are integrated ~~with,~~ are important considerations in their evolving design ~~as they evolve~~. ~~The~~ Students are not expected to create ~~creation of~~ integrated or embedded systems ~~is not an expectation~~ at this level.* *Practice(s):* Developing and Using Abstractions: 4.1
Subconcept: Hardware and Software (HS)	
HS.CS.HS.1	**Compare levels of abstraction and interactions between application software, system software, and hardware layers.** *At its most basic level, a computer is composed of physical hardware and electrical impulses. Multiple layers of software are built upon the hardware and interact with the layers above and below them to reduce complexity. System software manages a computing device's resources so that software can interact with hardware. For example, text editing software interacts with the*

	operating system to receive input from the keyboard, convert the input to bits for storage, and interpret the bits as readable text to display on the monitor. System software is used on many different types of devices, such as smart TVs, assistive devices, virtual components, cloud components, and drones. For example, students may explore the progression from voltage to binary signal to logic gates to adders and so on. Knowledge of specific, advanced terms for computer architecture, such as BIOS, kernel, or bus, is not expected at this level. *Practice(s):* Developing and Using Abstractions: 4.1
Subconcept: Troubleshooting (T)	
HS.CS.T.1	**Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.** *Troubleshooting complex problems involves the use of multiple sources when researching, evaluating, and implementing potential solutions. Troubleshooting also relies on experience, such as when people recognize that a problem is similar to one they have seen before ~~or~~ can adapt previous solutions ~~that have worked in the past~~. Examples of complex troubleshooting strategies include resolving connectivity problems, adjusting system configurations and settings, ensuring hardware and software compatibility, and transferring data from one device to another. Students could create a flow chart, a job aid for a help desk employee, or an expert system.* *Practice(s):* Testing and Refining Computational Artifacts: 6.2

Essential Concept: Networks and the Internet (NI)

Subconcept: Cybersecurity (C)	
HS.NI.C.1	**Give examples to illustrate how sensitive data can be affected by malware and other attacks.** *Network security depends on a combination of hardware, software, and practices that control access to data and systems. Potential security problems, such as denial-of-service attacks, ransomware, viruses, worms, spyware, and phishing~~,~~ present threats to sensitive data. Students might reflect on case studies or current events in which governments or organizations experienced data leaks or data loss as a result of these types of attacks.* *Practice(s): Communicating About Computing: 7.2*
HS.NI.C.2	**Recommend security measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.** *Security measures may include physical security tokens, two-factor authentication, and biometric verification. The timely and reliable access to data and information services by authorized users, referred to as availability, is ensured through adequate bandwidth, backups, and other measures. Students should systematically evaluate the feasibility of using computational tools (such*

	as a cost-benefit analysis) to solve given problems or subproblems, ~~such as through a cost-benefit analysis~~. Eventually, students should include more factors such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns into their evaluations, ~~such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns~~. *Practice(s): Recognizing and Defining Computational Problems: 3.3*
HS.NI.C.3	**Compare various security measures, considering tradeoffs between the usability and security of a computing system.** *Choosing security measures involves tradeoffs between the usability and security of the system. The needs of users and the sensitivity of data determine the level of security implemented. Students might discuss computer security policies in place at the local level that present a tradeoff between usability and security, such as a web filter that prevents access to many educational sites but keeps the campus network safe.* *Practice(s): Testing and Refining Computational Artifacts: 6.3*
Subconcept: Network, Communication, and Organization (NCO)	
HS.NI. NCO.1	**Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing.** *Each device is assigned an address that uniquely identifies it on the network. Routers function by comparing IP addresses to determine the pathways packets should take to reach their destinations. Switches function by comparing MAC addresses to determine which computers or network segments will receive frames. Students could use online network simulators to experiment with these factors.* *Practice(s): Developing and Using Abstractions: 4.1*

Essential Concept: Data and Analysis (DA)

Subconcept: Collection, Visualization and Transformation (CVT)	
HG.DA. CVT.1	**Create interactive data visualizations using software tools to help others better understand real-world phenomena.** *People use software tools or programming to create powerful, interactive data visualizations and perform a range of mathematical operations to transform and analyze data. Students should model phenomena as systems, with rules governing the interactions within these systems and evaluate these models against real-world observations.* *Practice(s): Developing and Using Abstractions: 4.4*
Subconcept: Storage (S)	

HS.DA.S.1	**Translate between different bit representations (such as characters, numbers, and images) of real-world phenomena~~, such as characters, numbers, and images~~.** *~~Understand that~~ Mm~~ost~~ computing systems use different numerical representations of non-numerical data. For example, convert hexadecimal color codes to decimal numbers, or represent characters in their ASCII/Unicode representation.* *Practice(s): Developing and Using Abstractions: 4.1*
HS.DA.S.2	**Evaluate ~~the~~ tradeoffs in how and where data is stored.** *People make choices about how and where data is stored. Students might consider the cost, speed, reliability, accessibility, privacy, and integrity tradeoffs between storing photo data on a mobile device versus in the cloud. Students should evaluate whether a chosen solution is most appropriate for a particular problem.* *Practice(s): Recognizing and Defining Computational Problems: 3.3*
Subconcept: Inference and Models (IM)	
HS.DA.IM.1	**Analyze computational models that represent relationships to help others better understand real-world phenomena.** *Computational models make predictions about processes or phenomenon based on selected data and features that can be represented in a spreadsheet or other organizational software. The amount, quality, and diversity of data and the features chosen can affect the quality of a model and ability to understand a system. Predictions or inferences are tested to validate models. Students should model phenomena as systems, with rules governing the interactions within the system. Students should analyze and evaluate these models against real-world observations.* *Practice(s): Developing and Using Abstractions: 4.4*

Essential Concept: Algorithms and Programming (AP)

Subconcept: Algorithms (A)	
HS.AP.A.1	**Create prototypes that use algorithms for practical intent, personal expression, or to address a societal issue** *A prototype is a computational artifact that demonstrates the core functionality of a product or process. Prototypes are useful for getting early feedback in the design process, and can yield insight into the feasibility of a product. 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. Students should develop computational artifacts that demonstrate the performance, reusability, and ease of implementation of an*

	algorithm in response to a task or a computational problem ~~that demonstrate the performance, reusability, and ease of implementation of an algorithm.~~ *Practice(s): Creating Computational Artifacts: 5.2*
Subconcept: Variables (V)	
HS.AP.V.1	**Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.** *Students should be able to identify common features in multiple segments of code and substitute a single segment that uses lists (arrays) to account for the differences.* *Practice(s): Developing and Using Abstractions: 4.1*
Subconcept: Control (C)	
HS.AP.C.1	**Justify the selection of specific control structures and explain the benefits and drawbacks of choices made~~,~~ when tradeoffs involve readability and program performance.** *Readability refers to how clear the program is to other programmers and can be improved through documentation. The discussion of performance is limited to a theoretical understanding of execution time; a quantitative analysis is not expected. Control structures at this level may include conditional statements, loops, event handlers, and recursion.* *Practice(s): Recognizing and Defining Computational Problems: 5.2*
HS.AP.C.2	Use events to initiate instructions ~~Design and iteratively develop~~in **computational artifacts** ~~for practical intent, personal expression, or to address a societal issue by using events to initiate instructions~~**.** *In this context, relevant computational artifacts include programs, mobile apps, or web apps. Events can be user-initiated, such as a button press, or system-initiated, such as a timer firing. At previous levels, students have learned to create and call procedures. Here, students design procedures that are called by events.* *Practice(s): Creating Computational Artifacts: 5.2*
Subconcept: Modularity (M)	
HS.AP.M.1	Use constructs such as procedures, modules, and/or objects to ~~Decompose~~ decompose **problems into** ~~smaller components~~subproblems.~~ through systematic analysis, using constructs such as procedures, modules, and/or objects.~~ *At this level, students should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist.* *Practice(s): Recognizing and Defining Computational Problems: 3.2*

Commented [MOU24]: There was too much going on in this standard and it obscured the primary learning outcome.

HS.AP.M.2	~~Create artifacts by using~~Use **procedures within a program, combinations of data and procedures, or independent but interrelated programs** in a computational artifact**.** *Computational artifacts can be created by combining and modifying existing artifacts or by developing new artifacts. Complex programs are designed as systems of interacting procedures, each with a specific role, coordinating for a common overall purpose. The focus at this level is understanding a program as a system with relationships between procedures.* *Practice(s): Creating Computational Artifacts: 5.2*
Subconcept: Program Development (PD)	
HS.AP.PD.1	**Evaluate and refine computational artifacts to make them more usable and accessible.** *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 should respond to the changing needs and expectations of end users and improve the performance, reliability, usability, and accessibility of artifacts. At this level, students should work through a systematic process that includes feedback from broad audiences.* *Practice(s): Testing and Refining Computational Artifacts: 6.3*
HS.AP.PD.2	~~Design and develop computational artifacts working in team roles~~Use team roles and ~~using~~ **collaborative tools** to design and develop computational artifacts**.** *Most software is developed in teams which can include pair programming or other collaborative structures. Team roles in pair programming are alternating driver and navigator but could be more specialized in larger teams. Students may choose to use collaborative tools to aid their team, such as a version control system or project management interface.* *Practice(s): Collaborating Around Computing: 2.1*
HS.AP.PD.3	~~Document design decisions using~~ Use **text, graphics, presentations, and/or demonstrations** to document design decisions when~~in the development~~ developing~~of~~ **complex programs.** *Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program; combinations of data and procedures; or independent, but interrelated, programs.* *Practice(s): Communicating About Computing: 7.2*

Essential Concept: Impacts of Computing (IC)

Subconcept: Culture (C)

HS.IC.C.1	**Evaluate the ways access to computing impacts personal, ethical, social, economic, and cultural practices.** *Computing may improve, harm, or maintain existing practices. Equity deficits, such as minimal exposure to computing, access to education, and training opportunities, are related to larger, systemic problems in society. Students should be able to evaluate the accessibility of a product to a broad group of end users, such as people who lack access to broadband or who have various disabilities.* *Practice(s): Fostering an Inclusive Computing Culture: 1.2*
HS.IC.C.2	**Test and refine computational artifacts to reduce bias and equity deficits.** *Biases could include incorrect assumptions developers have made about their user base or data. Students should begin to identify potential bias during the design process to maximize accessibility in product design, and become aware of professionally accepted accessibility standards to evaluate computational artifacts for accessibility.* *Practice(s): Fostering an Inclusive Computing Culture: 1.2*
HS.IC.C.3	**Demonstrate ways a given algorithm applies to problems across disciplines.** *Computation can share features with disciplines such as art and music by algorithmically translating human intention into an artifact. Students should be able to identify real-world problems that span multiple disciplines~~, such as increasing bike safety with new helmet technology,~~ and ~~that~~ can be solved computationally, such as increasing bike safety with new helmet technology.* *Practice(s): Recognizing and Defining Computational Problems: 3.1*
Subconcept: Social Interactions (SI)	
HS.IC.SI.1	**Analyze the impact of tools and methods for collaboration that increase connectivity.** *Many aspects of society, especially careers, have been affected by the degree of communication afforded by computing. The increased connectivity between people in different cultures and in different career fields has changed the nature and content of many careers. Students should explore different collaborative tools and methods used to solicit input from team members, classmates, and others, such as participation in online forums or local communities. For example, students could compare ways different social media tools could help a team become more cohesive.* *Practice(s): Collaborating Around Computing: 2.4*
Subconcept: Safety, Law, and Ethics (SLE)	
HS.IC.SLE.1	**Explain the beneficial and harmful effects that intellectual property laws can have on innovation.** *Laws govern many aspects of computing, such as privacy, data, property, information, and identity. These laws can have beneficial and harmful effects, such as expediting or delaying advancements in computing and protecting or infringing upon people's rights. International differences in laws and ethics have implications for computing. For example~~s~~, laws that mandate the blocking of some file-sharing websites may reduce online piracy but can restrict the right to access information. Students should be aware of*

	intellectual property laws and be able to explain how they ~~are~~ can be used to protect the interests of innovators or can ~~be~~ potentially be misused. *Practice(s): Communicating About Computing: 7.3*
HS.IC.SLE.2	**Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.** *Data can be collected and aggregated across millions of people, even when they are not actively engaging with or physically near the data collection devices. This automated and non-evident collection (such as social media sites mining an account even when the user is not online) can raise privacy concerns~~.~~* *~~such as social media sites mining an account even when the user is not online.~~* *Practice(s): Communicating About Computing: 7.2*
HS.IC.SLE.3	**Evaluate the social and economic implications of privacy in the context of safety, law, or ethics.** *Laws govern many aspects of computing, such as privacy, data, property, information, and identity. International differences in laws and ethics have implications for computing. Students might review case studies or current events which present an ethical dilemma when an individual's right to privacy is at odds with the safety, security, or wellbeing of a community.* *Practice(s): Communicating About Computing: 7.3*

Computer Science Glossary

The following glossary includes definitions of terms used in the statements in the K–12 Computer Science Framework. These terms are intended to increase teacher understanding and decrease biased language.

Commented [MOU25]: Although you are borrowing this glossary from the K-12 CS Framework, it is a bit confusing in this content. Suggest: "The following glossary includes definitions of commonly-used computer science concepts and was borrowed (with permission) from the K–12 Computer Science Framework."

Abstraction (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.

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

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 needed to make a peanut butter and jelly sandwich is an algorithm.

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 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: To link 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 method of encoding data using two symbols (usually 1 and 0). To illustrate binary encoding, we can use any two symbols.

A way of representing information using only two options.

Block-based programming language: Any programming language that lets users create programs by manipulating "blocks" or graphical programing elements, rather than writing code using text. Examples include Code Studio, Scratch, and Swift. (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 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 ~~(decomposing problems into smaller, more manageable problems, finding repeating patterns, abstracting specific differences to make one solution work for multiple problems, and creating~~ (a step-by-step ~~algorithms~~set of instructions that can be acted upon by a computer).

Computer science: ~~Using the power of computers to solve problems.~~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.

Commented [MOU26]: I know you are trying to provide a selection of examples but most standards documents tend to avoid including vendor products. Might be better just to use Code Studio and Scratch since they are not vendor-created.

Commented [MOU27]: This is the ACM definition of the discipline and is the one most commonly-cited in CS education.

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. (Code.org K–5)

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

For loop: A loop with a predetermined beginning, end, and increment (step interval) (Code.org K–5)

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~~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 man-in-the-middle attacks.

Input: The signals or instructions sent to a computer.

Commented [MOU28]: Not sure you should include this as this term also requires a definition and I am not sure you need it.

Internet: The global collection of computer networks and their connections, all using shared protocols to communicate [CAS-Prim]. 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: Ddata compression without loss of information.

Lossy: Ddata 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.

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.

Nnested Lloop: A loop within a loop, an inner loop within the body of an outer one.

Formatted: Font: Bold

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~~; programming (~~(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.

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 ~~you've~~ written in ~~your~~ 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: T~~t~~o 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~~the~~ 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: A general term ~~used in the framework to discuss~~to describe 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 ~~is~~ details how devices appear connected to the user. A physical topology ~~is~~ details how ~~they~~ 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.

Wearable computing: Miniature electronic devices that are worn under, with or on top of clothing. Note: This definition differs from that used in math.

Commented [MOU29]: Not sure this is even a concept in mathematics.

References:

Numerous existing sets of standards and standards-related documents have been used in developing the Arizona Standards for Computer Science. 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/
- ISTE Standards, 2016: www.iste.org/standards/for-students#startstandards
- College Board's Computer Science Principles: https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf
- 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
    - South Carolina Computer Science and Digital Literacy Standards: https://ed.sc.gov/state-board/state-board-of-education/full-state-board-meeting/full-board-sbe/may-2017/secondreading-attachment-proposed-sc-computer-science-amp-digital-literacy-standards-pdf/
- The following publication(s) were referenced:
    - Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2016). A model curriculum for K-12 computer science: Report of the ACM K-12 task force curriculum committee (2nd ed.). New York, NY: Association for Computing Machinery.
- The following website(s) were used for reference:
    - 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
    - FOLDOC: Free On-Line Dictionary of Computing: http://foldoc.org/
    - MA-DLCS: Massachusetts Digital Literacy and Computer Science Standards, Glossary (Draft, December 2015): http://www.doe.mass.edu/frameworks/dlcs.pdf

- 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