Arizona Computer Science Standards

Third Grade – Fifth Grade

ARIZONA DEPARTMENT OF EDUCATION

HIGH ACADEMIC STANDARDS FOR STUDENTS

Adopted October 2018

Contents

Vision Statement

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

Introduction

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

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

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

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

Focus On Equity

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

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

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

Acknowledgements

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

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

Computer Science Essential Concepts and Subconcepts

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

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

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

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

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

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

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

Figure 1: Computer science essential concepts and subconcepts

Computing Systems	**Networking and the Internet**	**Data and Analysis**	**Algorithms and Programming**	**Impacts of Computing**
Devices	*Network Communication and Organization*	*Collection*	*Algorithms*	*Culture*
Hardware and Software	*Cybersecurity*	*Storage*	*Variables*	*Social Interactions*
Troubleshooting		*Visualization and Transformation*	*Control*	*Safety, Law, and Ethics*
		Inference and Models	*Modularity*	
			Program Development	

Computer Science Practices for Students

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

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

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

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

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

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

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

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

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

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

Figure 2: Essential Practices including computational thinking

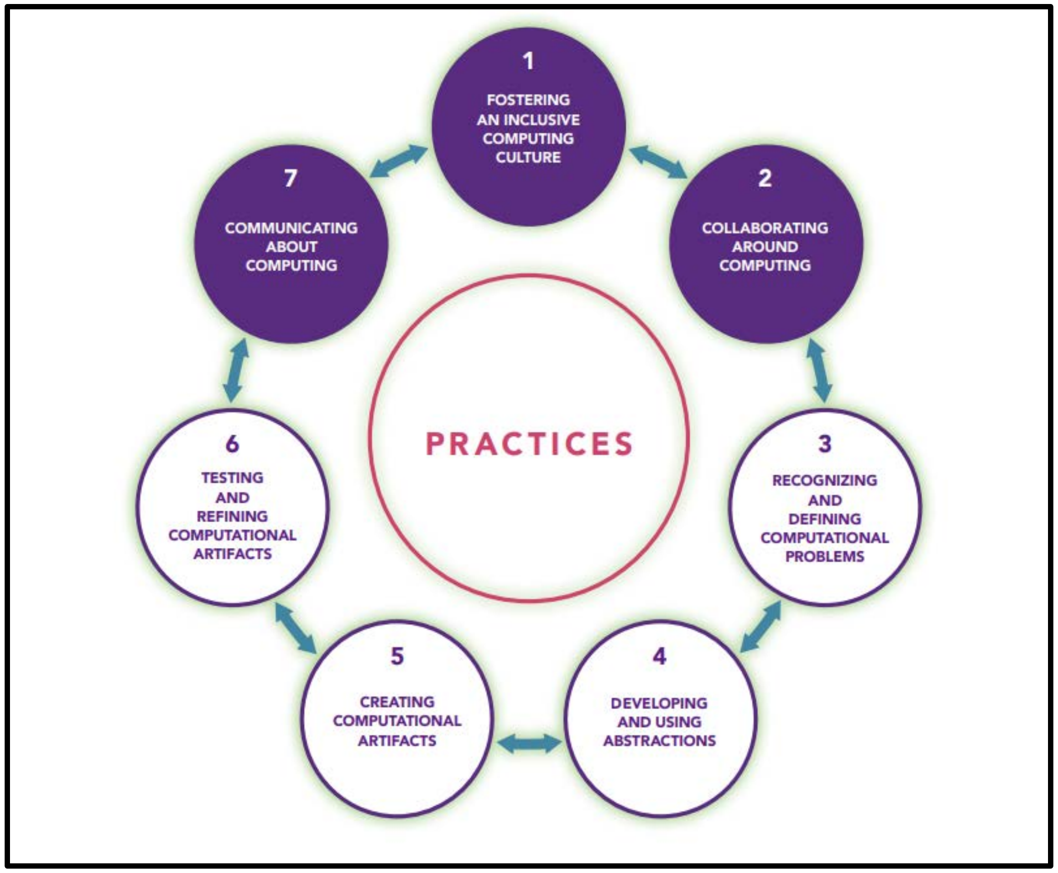
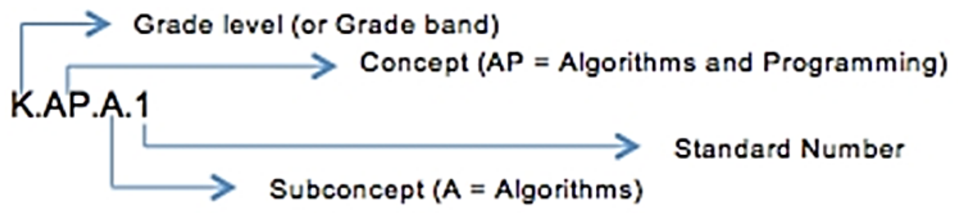

1
FOSTERING AN INCLUSIVE COMPUTING CULTURE
2
COLLABORATING AROUND COMPUTING
3
RECOGNIZING AND DEFINING COMPUTATIONAL PROBLEMS
4
DEVELOPING AND USING ABSTRACTIONS
5
CREATING COMPUTATIONAL ARTIFACTS
6
TESTING AND REFINING COMPUTATIONAL ARTIFACTS
7
COMMUNICATING ABOUT COMPUTING
PRACTICES

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

How to Read the Arizona Computer Science Standards

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

Figure 3: Standard Coding Scheme for Standards


Grade level (or Grade band)
Concept (AP = Algorithms and Programming)
K.AP.A.1
Standard Number
Subconcept (A = Algorithms)

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

Figure 4: Example of Complete Individual Standard

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

Third Grade

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

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 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 (devices) and software (programs/apps) communicate in a special language that the computing system can understand.** *Computing systems convert instructions, such as "print," "save," or "crop," into a special language that the computer can understand. Students discuss the process that happens when hardware communicates with software.* *Practice(s): Communicating About Computing: 7.2*
3.CS.HS.2	**Recognize that hardware (devices) can only accomplish the specific tasks the software (programs/apps) is designed to accomplish.** *Cameras can take pictures because the camera software allows them to do so. Students discuss examples of different hardware and the tasks they can accomplish.* *Practice(s): Communicating About Computing: 7.2*

Subconcept: Troubleshooting (T)	
3.CS.T.1	**Identify and use 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. Restarting a computer (rebooting) is commonly effective because it resets the machine. 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*

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 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 such as adding passwords to lock devices.* *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 through a physical or wireless path.** *There are physical paths for communicating information, such as Ethernet cables, and wireless* paths (Wifi). *Often, information travels on a combination of physical and wireless paths. Wireless paths originate from a physical connection point and travel through multiple devices and wired or wireless connections to their end point. Models could include visual, physical, or alternate representations.* *Practice(s): Developing and Using Abstractions: 4.3*

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. Students discuss common file extensions, such as .doc, .pdf, and .jpeg.* *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. A computational tool can be anything used or analyzed to draw conclusions, make predictions, or answer questions.* *Practice(s): Communicating about Computing, Collaborate around Computing: 7.2, 2.4*

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 effective.** *Different algorithms can achieve the same result, though sometimes one algorithm might be most appropriate for a specific task. Students 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. Students may use mathematical operations to add to the score of a game or subtract from the number of lives in a game. Programs can imply either digital or 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. If dialogue is not sequenced correctly when programming a simple animated story, the story will not make sense. Events allow portions of a program to run based on a specific action. 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. 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. 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 problems into smaller, manageable subproblems to facilitate the program development process.** *Decomposition is the act of breaking down 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
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. With teacher guidance, 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 the creator may not like. Students should identify if ideas were borrowed or adjusted 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. Identifying a mistake in a math problem, for example; Sally solved the following problem as 11, there were five groups with six apples in each. How many apples were there? Was she correct? Fix her mistake if she was incorrect.* Practice(s): Testing and Refining Computational Artifacts: 6.1, 6.2
3.AP.PD.4	With teacher guidance, students take on varying roles, **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 (procedure) 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. Students work together to explain the steps in their procedure.* Practice(s): Communicating About Computing: 7.2

Concept: Impacts of Computing (IC)

Subconcept: Culture (C)	
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. With guidance from their teacher, 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
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 their own tastes.* Practice(s): Fostering an Inclusive Computing Culture: 1.2
Subconcept: Social Interactions (SI)	
3.IC.SI.1	**Seek opportunities for local 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 or pen pals.* Practice(s): Fostering an Inclusive Computing Culture: 1.1
Subconcept: Safety, Law, and Ethics (SLE)	
3.IC. SLE.1	**Use material that is publicly available and/or permissible to use.** *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, creates the opportunity for unauthorized use, such as online piracy and disregard of copyrights. Students should consider the licenses 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, the computer literate student refines their skills as they construct programs and use algorithms to accomplish a task. Analyzing a variety of hardware and software tools, students 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 protecting personal information and respecting the rights of others.

Concept: Computing Systems (CS)

Subconcept: Devices (D)	
4.CS.D.1	**With teacher guidance, model how internal and external parts of computing** *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. Students model the process that happens when multiple devices form a system* *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. Students discuss how bits are a unit of data.* *Practice(s): Communicating About Computing: 7.2*
4.CS.HS.2	**Recognize that a single piece of hardware can accomplish different tasks depending on** *its* **software.** *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. Students discuss a variety of software and hardware that work together.* *Practice(s): Practice(s): Communicating About Computing: 7.2*

Subconcept: Troubleshooting (T)	
4.CS.T.1	**Develop** *and apply simple 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. Restarting a device (rebooting) is commonly effective because it resets the 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*

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 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: 7.2, 3.3*
Subconcept: Network, Communication, and Organization (NCO)	
4.NI. NCO.1	**Model how information is decomposed, transmitted as packets through multiple devices over networks 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. Information is broken down into smaller pieces called packets, which are sent over the network 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*

Concept: Data and Analysis (DA)

Subconcept: Collection, Visualization and Transformation (CVT)	
4.DA. CVT.1	**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.* *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 the different amounts of storage required for each type.** *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. Students discuss common file extensions, such as .doc, .pdf, and .jpeg and give examples of files that require different amounts of 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*

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 effective.** 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 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** *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. Students may use mathematical operations to add to the score of a game or subtract from the number of lives in a game. Programs can imply either digital or 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. If dialogue is not sequenced correctly when programming a simple animated story, the story will not make sense. Events allow portions of a program to run based on a specific action. 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. 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. 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 problems into smaller, manageable subproblems to facilitate the program development process.** *Decomposition is the act of breaking down 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
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 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/app 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	**With teacher guidance, students take on varying roles 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

Concept: Impacts of Computing (IC)

Subconcept: Culture (C)	
4.IC.C.1	**Identify and discuss computing technologies that have changed the world.** *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. Students discuss how culture influences changes in 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 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 nationally 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 material that is publicly available and/or permissible to use.** *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, creates the opportunity for unauthorized use, such as online piracy and disregard of copyrights. Students should consider the licenses 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

The computer literate student independently and collaboratively constructs programs and uses algorithms to accomplish real world tasks. Students continue to decompose larger problems into smaller tasks, recognize the impacts of computing and computing devices and model how computing systems work. The accurate use of terminology as well as the responsible use of technology will continue with emphasis on intellectual property rights.

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. 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 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	**Explain how computing devices affect humans in positive and negative ways.** *The use of computing devices has potential consequences, 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. Information is transformed into binary digits, for example a song is stored as more binary digits than a photo.* *Practice(s): Communicating About Computing, Creating Computational Artifacts: 7.2, 5.2*
5.CS.HS.2	**Demonstrate and explain 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 a animation to demonstrate it, or demonstrate it by acting it out in some way. *Practice(s): Communicating About Computing, Creating Computational Artifacts: 7.2, 5.3*

Subconcept: Troubleshooting (T)	
5.CS.T.1	**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 in power or swapping a working part in place of a potentially defective part can be used to restore functionality. Restarting a device (rebooting) is commonly effective because it resets the computer machine. Computing devices are composed of an interconnected system of hardware and software, troubleshooting strategies may need to address both. In fifth grade students begin troubleshooting complex problems through networks, routers, and switches.* *Practice(s): Recognizing and Defining Computational Problems, Developing and Using Abstractions: 3.2, 4.1*

Concept: Networks and the Internet (NI)

Subconcept: Cybersecurity (C)	
5.NI.C.1	**Identify solutions to 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: 7.2, 3.1*
Subconcept: Network, Communication, and Organization (NCO)	
5.NI. NCO.1	**Analyze 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*

Concept: Data and Analysis (DA)

Subconcept: Collection, Visualization and Transformation (CVT)	
5.DA. CVT.1	**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	**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 propose cause-and-effect relationships, predict outcomes, or communicate an idea.** *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*

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 effective.** 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	**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 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 designs. Students* create programs that use variables to store and modify data. *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. 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: 5.1
Subconcept: Modularity (M)	
5.AP.M.1	**Decompose problems into manageable subproblems to facilitate the program development process.** *Decomposition is the act of breaking down a task into multiple, simpler tasks. Decomposition also enables different people to work on different parts at the same time. 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.** *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.** *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 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.** *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.** *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.** *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

Concept: Impacts of Computing (IC)

Subconcept: Culture (C)	
5.IC.C.1	**Discuss computing technologies that have changed the world.** *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. Students discuss how culture influences changes in 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, cultural and political changes.* Practice(s): Recognizing and Defining Computational Problems: 3.1
5.IC.C.2	**Design 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 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, creates the opportunity for unauthorized use, such as online piracy and disregard of copyrights. Students should consider the licenses 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

Appendix A-Computer Science Glossary

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

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

- (Process): The process of reducing complexity by focusing on the main idea. By hiding details irrelevant to the question at hand and bringing together related and useful details, abstraction reduces complexity and allows one to focus on the problem. In elementary classrooms, abstraction is hiding unnecessary details to make it easier to think about a problem.
- (Product): A new representation of a thing, a system, or a problem that helpfully reframes a problem by hiding details irrelevant to the question at hand.

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

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

App: A type of application software, often designed to run on a mobile device, such as a smartphone or tablet computer (also known as a mobile application).

Artifact: Anything created by a human. See “computational artifact” for the computer science-specific definition.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Debugging: Finding and fixing errors in programs.

Decompose: Break a problem down into smaller pieces.

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

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

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

Event: An action that causes something to happen

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

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

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

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

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

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

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

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

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

Input: The signals or instructions sent to a computer.

Internet: The global collection of computer networks and their connections, all using shared protocols to communicate. A group of computers and servers that are connected to each other.

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

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

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

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

Lossless: Data compression without loss of information.

Lossy: Data compression in which unnecessary information is discarded.

Memory: Temporary storage used by computing devices.

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

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

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

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

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

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

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

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

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

Parameter: A special kind of variable used in a procedure to refer to one of the pieces of data provided as input to the procedure. These pieces of data are called arguments. An ordered list of parameters is usually included in the definition of a subroutine so each time the subroutine is called, its arguments for that call can be assigned to the corresponding parameters. An extra piece of information that you pass to a function to customize it for a specific need.

Pattern matching: Finding similarities between things.

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

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

Procedure: An independent code module that fulfills some concrete task and is referenced within a larger body of source code. This kind of code item can also be called a function or a subroutine. The fundamental role of a procedure is to offer a single point of reference for some small goal or task that the developer or programmer can trigger by invoking the procedure itself. A procedure may also be referred to as a function, subroutine, routine, method, or subprogram.

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

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

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

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

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

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

Remix: making changes to an existing procedure.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

System: A collection of elements or components that work together for a common purpose. A collection of computing hardware and software integrated for the purpose of accomplishing shared tasks.

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

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

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

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

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

Sources for definitions in this glossary:

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

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

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

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

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

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

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

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

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

Appendix B-Computer Science Practices

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

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

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

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