Public Schools of North Carolina
State Board of Education | Department of Public Instruction

2020 North Carolina K12 Computer Science Standards with Descriptions.

This document is designed to help North Carolina educators teach the NC Standard Course of Study for Computer Science.

This document provides more detailed descriptions of each standard in the 2020 NC K12 Computer Science Standards which are based on the 2017 Computer Science Teachers Association Computer Science Standards.

Kindergarten through Second Grade

K2-CS-01	**Choose appropriate devices to perform a variety of classroom tasks.**	**Computing Systems** Devices
	People use computing devices to perform a variety of tasks accurately and quickly. 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 record video or audio, draw a picture, they should be able to identify devices with these capabilities.	
K2-CS-02	**Describe the function of common physical components of computing systems (hardware) with appropriate terminology.**	**Computing Systems** Hardware & Software
	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.	
K2-CS-03	**Operate appropriate software to perform a variety of tasks.**	**Computing Systems** Hardware & Software
	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.	
K2-CS-04	**Describe basic hardware and software problems with accurate terminology.**	**Computing Systems** Troubleshooting
	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.). 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.	
K2-NI-01	**Illustrate how information is broken down into smaller pieces and can be reassembled.**	**Networks & the Internet** Network Communication & Organization

	When information is sent and received over physical or wireless paths, it is broken down into smaller pieces called packets, which are sent independently and reassembled at the destination. Students should demonstrate their understanding of this concept through an activity which has them act it out in some way, e.g. taking apart a puzzle and reassembling it.	
K2-NI-02	**Apply knowledge of what passwords are and why we use strong passwords to protect devices and information from unauthorized access.**	**Networks & the Internet** Cybersecurity
	Learning to protect one's device or information from unwanted use by others is an essential first step in learning about cybersecurity. Students are not required to use multiple strong passwords. They should appropriately use and protect the passwords they are required to use.	
K2-NI-03	**Discover your digital footprint and how personal information can be protected.**	**Networks & the Internet** Cybersecurity
	Information stored on a computer or the Internet needs protecting, just as we protect our personal property offline. Ways to protect our digital property can include backing up data to guard against loss, creating and not sharing strong passwords, or installing anti-virus software.	
K2-DA-01	**Store, copy, search, retrieve, modify, and delete information using a computing device.**	**Data & Analysis** Storage
	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. As students use software to complete tasks on a computing device, they will be manipulating data.	
K2-DA-02	**Define information stored on a computing device as data.**	**Data & Analysis** Storage
	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. As students use software to complete tasks on a computing device, they will be manipulating data.	
K2-DA-03	**Collect and present the same data in various visual formats.**	**Data & Analysis** Collection Visualization & Transformation
	The collection and use of data about the world around them is a routine part of life and influences how people live. 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.	
K2-DA-04	**Make predictions with patterns in data visualizations.**	**Data & Analysis** Inference & Models
	Data can be used to make inferences or predictions about the world. Students could analyze a graph or 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.	
K2-AP-01	**Model daily processes with algorithms to complete tasks.**	**Algorithms & Programming** Algorithms
	Composition is the combination of smaller tasks into more complex tasks. Students could create and follow algorithms for making simple foods, brushing their teeth, getting ready for school, participating in clean-up time.	
K2-AP-02	**Demonstrate how programs store and manipulate data by using numbers or other symbols to represent information.**	**Algorithms & Programming** Variables
	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.	
K2-AP-03	**Develop programs with sequences and simple loops to express ideas or address a problem.**	**Algorithms & Programming** Control
	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. Sequences are the order of instructions in a program. 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 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.	

K2-AP-04	**Decompose the steps needed to solve a problem into a precise sequence of instructions.**	**Algorithms & Programming** Modularity
	Decomposition is the act of breaking down tasks into simpler tasks. Students could break down the steps needed to make a peanut butter and jelly sandwich, to brush their teeth, to draw a shape, to move a character across the screen, or to solve a level of a coding app.	
K2-AP-05	**Develop plans that describe a program's sequence of events, goals, and expected outcomes.**	**Algorithms & Programming** Program Development
	Creating a plan for what a program will do clarifies the steps that will be needed to create a program and can be used to check if a program is correct. 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.	
K2-AP-06	**Give attribution when using the ideas and creations of others while developing programs.**	**Algorithms & Programming** Program Development
	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.	
K2-AP-07	**Identify and debug errors in an algorithm or program that includes sequences and simple loops.**	**Algorithms & Programming** Program Development
	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.	
K2-AP-08	**Using correct terminology, describe steps taken and choices made during the iterative process of program development**	**Algorithms & Programming** 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.	
K2-IC-01	**Compare how people live and work before and after the implementation or adoption of new computing technology.**	**Impacts of Computing** Culture
	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.	
K2-IC-02	**Select software that meets the diverse needs and preferences for the technology individuals use in the classroom.**	**Computing Systems** Culture
	People use computing devices to perform a variety of tasks accurately and quickly. 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.	
K2-IC-03	**Work respectfully and responsibly with others online.**	**Impacts of Computing** Social Interactions
	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 and could tell an adult if others are sharing things they should not share or are treating others in an unkind or disrespectful manner on online collaborative spaces.	
K2-IC-04	**Model responsible login and logoff procedures on all devices.**	**Impacts of Computing** Safety Law & Ethics
	Protecting our information online is a very important part of being a citizen of the digital world. We have a role in protecting our information and identity online. We should realize that the process of logging on and off a device is a critical step in this process.	

Third through Fifth Grade

35-CS-01	**Evaluate the features available on digital devices to perform a variety of classroom tasks.**	**Computing Systems** Devices
	As students are exposed to more and varied devices, they should begin to recognize the strengths and weaknesses of these devices. Students should be able to determine the appropriateness of a device for a given task based on its features. For example, which devices might be ideal	

	for drawing, capturing video or audio, and editing media. Students should also understand how to use different devices to complete a given task, where some work is done on a simple device and then moved to a more capable device.	
35-CS-02	**Model how computer hardware and software work together as a system to accomplish tasks.**	**Computing Systems** Hardware & 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 out in some way.	
35-CS-03	**Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies.**	**Computing Systems** Troubleshooting
	Although computing systems may vary, common troubleshooting strategies can be used on all of them. Students should be able to identify solutions to problems such as the device not responding, no power, no network, app crashing, no sound, or password entry not working. Should errors occur at school, the goal would be that students would use various strategies, such as rebooting the device, checking for power, checking network availability, closing and reopening an app, making sure speakers are turned on or headphones are plugged in, and making sure that the caps lock key is not on, to solve these problems, when possible.	
35-NI-01	**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.**	**Networks & the Internet** Network Communication & Organization
	Information is sent and received over physical or wireless paths. It is broken down into smaller pieces called packets, which are sent independently and reassembled at the destination. Students should demonstrate their understanding of this flow of information by, for instance, drawing a model of the way packets are transmitted, programming an animation to show how packets are transmitted, or demonstrating this through an unplugged activity which has them act it out in some way.	
35-NI-02	**Explain your digital footprint and how personal information can be protected.**	**Networks & the Internet** Cybersecurity
	Just as we protect our personal property offline, 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. Students could discuss or use a journaling or blogging activity to explain, orally or in writing, about topics that relate to personal data security issues. Discussion topics could be based on current events related to cybersecurity or topics that are applicable to students, such as the necessity of backing up data to guard against	

	loss, how to create strong passwords and the importance of not sharing passwords, or why we should install and keep anti-virus software updated to protect data and systems.	
35-DA-01	**Identify the type of data encoded in a file based on file extension.**	**Data & Analysis** Storage
	Before information can be stored on a computing device, the information must be converted, encoded. In many computing systems, the way information is stored depends on the type of information (text, images, movies, ...). In many cases, you can determine the type of information from the file name extension (.DOC, .GIF, .MOV) Teachers could use word walls or other classroom decorations and label them with file extensions.	
35-DA-02	**Illustrate the process of file management and version control.**	**Data & Analysis** Storage
	As we work in an ever more digital world, managing digital clutter is critical. Students should understand how files are stored on various devices and using various operating systems. In addition, students should manage progress on projects using a system to indicate versions and iterations.	
35-DA-03	**Organize and present collected data visually to highlight relationships and support a claim.**	**Data & Analysis** Collection Visualization & Transformation
	Raw data has little meaning on its own. Data is often sorted or grouped to provide additional clarity. 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 in different ways to emphasize particular aspects or parts of the data set. For example, a data set of sports teams could be sorted by wins, points scored, or points allowed, and a data set of weather information could be sorted by high temperatures, low temperatures, or precipitation.	
35-DA-04	**Communicate using data to highlight or predict outcomes.**	**Data & Analysis** Inference & Models
	The accuracy of data analysis is related to how realistically data is represented. Inferences or predictions based on data are less likely to be accurate if the data is not sufficient or if the data is incorrect in some way. Students should be able to refer to data when communicating an idea. For example, in order to explore the relationship between speed, time, and distance, students could operate a robot at uniform speed, and at increasing time intervals to predict how far the robot travels at that speed. In order to make an accurate prediction, one or two attempts of differing times would not be enough. The robot may also collect temperature data from a sensor, but that data would not be	

	relevant for the task. Students must also make accurate measurements of the distance the robot travels in order to develop a valid prediction. Students could record the temperature at noon each day as a basis to show that temperatures are higher in certain months of the year. If temperatures are not recorded on non-school days or are recorded incorrectly or at different times of the day, the data would be incomplete and the ideas being communicated could be inaccurate. Students may also record the day of the week on which the data was collected, but this would have no relevance to whether temperatures are higher or lower. In order to have sufficient and accurate data on which to communicate the idea, students might want to use data provided by a governmental weather agency.	
35-AP-01	**Create multiple algorithms for the same task to determine which is the most accurate and efficient.**	**Algorithms & Programming** Algorithms
	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.	
35-AP-02	**Create programs that use variables to store and modify data.**	**Algorithms & Programming** Variables
	Variables are used to store and modify data. At this level, understanding how to use variables is sufficient. For example, students may use mathematical operations to add to the score of a game or subtract from the number of lives available in a game. The use of a variable as a countdown timer is another example.	
35-AP-03	**Construct programs that include sequences.**	**Algorithms & Programming** Control
	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.	
35-AP-04	**Construct programs using simple loops.**	**Algorithms & Programming** Control
	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.	

35-AP-05	**Construct programs that implement conditionals.**	**Algorithms & Programming** Control
	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.	
35-AP-06	**Decompose problems into smaller, manageable, subproblems to facilitate the program development process.**	**Algorithms & Programming** Modularity
	Decomposition is the act of breaking down tasks into simpler tasks. 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.	
35-AP-07	**Modify, remix, or incorporate portions of an existing program into one's own work.**	**Algorithms & Programming** Modularity
	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.	
35-AP-08	**Apply an iterative process to the development of a program by including diverse perspectives and considering user preferences.**	**Algorithms & Programming** Program Development
	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.	
35-AP-09	**Give appropriate attribution when creating or remixing programs while respecting intellectual property rights.**	**Algorithms & Programming** Program Development
	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 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.	

35-AP-10	**Identify and debug errors in an algorithm or program to ensure it runs as intended.**	**Algorithms & Programming** Program Development
	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.	
35-AP-11	**Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development.**	**Algorithms & Programming** 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.	
35-AP-12	**Describe choices made during program development using code comments, presentations, and demonstrations.**	**Algorithms & Programming** Program Development
	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.	
35-IC-01	**Compare computing technologies that have changed the world and how they both influence and are influenced by cultural practices.**	**Impacts of Computing** Culture
	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, such as robotics, wireless Internet, mobile computing devices, GPS systems, wearable computing, or how social media has influenced social and political changes. Students should compare the relative impact of various innovations.	
35-IC-02	**Explore the tools that can be used to improve accessibility and usability of technology products for the diverse needs and wants of users.**	**Impacts of Computing** Culture
	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.	
35-IC-03	**Seek diverse perspectives with collaboration for the purpose of improving computational artifacts.**	**Impacts of Computing** Social Interactions
	Computing provides the possibility for collaboration and sharing of ideas and allows the benefit of diverse perspectives. For example, students could seek feedback from other groups in their class or students at another grade level. Or, with guidance from their teacher, they could use video conferencing tools or other online collaborative spaces, such as blogs, wikis, forums, or website comments, to gather feedback from individuals and groups about programming projects.	
35-IC-04	**Exhibit positive digital citizenship and social responsibility.**	**Impacts of Computing** Social Interactions
	Students should recognize that the digital world is shared, and as with the physical world we inhabit, we each have a role in keeping the world safe and healthy.	
35-IC-05	**Utilize public domain or creative commons media, and refrain from copying or using material created by others without permission.**	**Impacts of Computing** Safety Law & Ethics
	Ethical complications arise from the opportunities provided by computing. The ease of sending and receiving copies of media 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 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.	

Sixth through Eighth Grade

68-CS-01	**Understand the design of computing devices based on an analysis of how users interact with the**	**Computing Systems** Devices
	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.	
68-CS-02	**Design projects that combine hardware and software components to collect and exchange data.**	**Computing Systems** Hardware & Software
	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 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.	
68-CS-03	**Systematically identify and fix problems with computing devices and components.**	**Computing Systems** Troubleshooting
	Since a computing device may interact with interconnected devices within a system, problems may not be due to the specific computing device itself but to 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.	
68-NI-01	**Analyze different ways that data is transferred across a network and the role of protocols in transmitting data.**	**Networks & the Internet** Network Communication & Organization
	Protocols are rules that define how messages between computers are sent. They determine how quickly and securely information is transmitted across networks and the Internet, as well as how to handle errors in transmission. Students should analyze different ways 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 grade 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.	
68-NI-02	**Explain how physical and digital security measures protect electronic information.**	**Networks & the Internet** Cybersecurity
	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.	
68-NI-03	**Explain permission and authorizations to access resources to computer systems online.**	**Networks & the Internet** Cybersecurity
	As more computing resources move online, students should understand how users gain access to protected content through permissions and authorizations. Students should be able to explain the various roles of passwords, two-factor authentication, and tokens. They should demonstrate knowledge of the trade-offs for these methods.	
68-NI-04	**Apply multiple methods of encryption to model the secure transmission of information.**	**Networks & the Internet** Cybersecurity
	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.	
68-DA-01	**Represent data using multiple encoding schemes.**	**Data & Analysis** Storage
	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).	
68-DA-02	**Collect data using computational tools.**	**Data & Analysis**

		Collection Visualization & Transformation
	Our digital world is driven by data. Students should develop the necessary skills to identify and collect data. From data collected in a scientific experiment to data gathered from online resources, students should be able to use computational tools to collect and organize the data.	
68-DA-03	**Transform the collected data to make it more useful and**	**Data & Analysis** Collection Visualization & Transformation
	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 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.	
68-DA-04	**Refine computational models based on the data they have generated and/or data collected.**	**Data & Analysis** Inference & Models
	A model may be a programmed simulation of events or a representation of how various data is related. In order 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.	
68-AP-01	**Implement flowcharts and/or pseudocode to address complex problems as algorithms.**	**Algorithms & Programming** 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 recommendation algorithm and to identify other inputs they may have initially excluded.	

68-AP-02	**Create clearly named variables that represent different data types.**	**Algorithms & Programming** Variables
	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.	
68-AP-03	**Design and iteratively develop programs that combine control structures including nested loops and compound conditionals.**	**Algorithms & Programming** Control
	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.	
68-AP-04	**Construct programs that include events.**	**Algorithms & Programming** Control
	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.	
68-AP-05	**Organize problems and subproblems into parts.**	**Algorithms & Programming** Modularity
	Students should break down problems into subproblems, which can be further broken down to smaller parts. Decomposition facilitates aspects of program development by allowing students to focus on one piece at a time (e.g., getting input from the user, processing the data, and displaying the result to the user). Decomposition also enables different students to work on different parts at the same time. For example, animations can be decomposed into multiple scenes, which can be developed independently.	
68-AP-06	**Explain the design, implementation, and review of programs**	**Algorithms & Programming** Modularity
	In order to understand how programs and applications evolve into a viable product, students should be able to explain components of the development lifecycle, including design, implementation, and review.	

68-AP-07	**Create procedures with parameters to organize code and make it easier to reuse groups of instructions.**	**Algorithms & Programming** Modularity
	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.	
68-AP-08	**Assess feedback from team members and users to refine a solution that meets user needs.**	**Algorithms & Programming** Program Development
	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.	
68-AP-09	**Incorporate existing code and media into original programs and give attribution.**	**Algorithms & Programming** Program Development
	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 portions of code that create a realistic jump movement from another person's game, and they may also import Creative Commons-licensed images to use in the background. Students should give attribution to the original creators to acknowledge their contributions.	
68-AP-10	**Systematically test and refine programs using a range of test cases.**	**Algorithms & Programming** Program Development
	Use cases and test cases are created and analyzed to better meet the needs of users and to evaluate whether programs function as intended. 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).	
68-AP-11	**Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.**	**Algorithms & Programming** Program Development

	Collaboration is a common and crucial practice in programming 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.	
68-AP-12	**Document programs in order to make them easier to follow, test, and debug.**	**Algorithms & Programming** Program Development
	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 their product and communicate their process using design documents, flowcharts, and presentations.	
68-IC-01	**Compare tradeoffs associated with computing technologies that affect everyday activities and career options.**	**Impacts of Computing** Culture
	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.	
68-IC-02	**Describe how equity, access, and influence impact the distribution of computing resources in a global society.**	**Impacts of Computing** Culture
	The distribution of computing resources is not uniform across all parts of our global society. Students should be able to describe how equity and access are impacted by factors such as wealth, geographic location, governmental programs.	
68-IC-03	**Discuss issues of bias and accessibility in the design of existing technologies.**	**Impacts of Computing** Culture
	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 lighter 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.	

68-IC-04	**Collaborate, model, and promote effective research strategies for assessing and evaluating innovative resources.**	**Impacts of Computing** Culture
	While the Internet has become an incredible source for information, it has also become overwhelming as a means for identifying meaningful resources. Students should develop the necessary research skills to be able to identify and evaluate appropriate resources on the Internet, including those that demonstrate innovative approaches to problems and their solutions.	
68-IC-05	**Collaborate with many contributors to create a computational artifact.**	**Impacts of Computing** Social Interactions
	In the digital world, collaboration is not defined by simply the people physically in a room or even virtually on the same team. Utilizing online resources, students can take collaboration to a much larger scale throug, crowdsourcing, the gathering 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, like Scratch and Minecraft). For example, a group of students could combine animations to create a digital community mosaic. They could also solicit feedback from many people through use of online communities and electronic surveys.	
68-IC-06	**Utilize tools and methods for collaboration on a project to increase connectivity of peers.**	**Impacts of Computing** Social Interactions
	Many aspects of society, especially careers, have been affected by the degree of communication afforded by computing. 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.	
68-IC-07	**Examine the benefits and drawbacks of a digital footprint and online identity**	**Impacts of Computing** Social Interactions
	As students build their digital footprint and online identity, they should be aware of the potential benefits and drawbacks these create. While students likely appreciate the social benefit of feely connected online and the sense of accomplishment when publishing an artifact online, they should also realize that their online presence may also create potential disadvantages. Students should examine how others may access their information, the permanence of digital artifacts, and the ability of others to distort this information. For example, students might want to explore online reviews of restaurants, high profile celebrities and the impact of rumors, and current events that highlight social media pitfalls.	
68-IC-08	**Understand how online interactions make an impact on the social, emotional, and physical aspect of others**	**Impacts of Computing** Social Interactions

	The Internet is a powerful public square and represents the most connected space in the history of mankind. As such, students should know that interactions online likely have more impact than those in person. Actions online have a chance to grow disproportionately and may cause significant impact on others.	
68-IC-09	**Compare tradeoffs between allowing information to be public and keeping information private and secure.**	**Impacts of Computing** Safety Law & Ethics
	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.	
68-IC-10	**Explore how laws and regulations impact the development and use of software**	**Impacts of Computing** Safety Law & Ethics
	Our society is governed by laws and regulations. Students should explore how these policies impact how software and technology can evolve. For example, students might explore how antitrust and monopoly regulation have impacted companies from Bell Systems to Microsoft to Amazon.	

Introductory Computer Science

ICS-CS-01	**Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.**	**Computing Systems** Devices
	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 creation of integrated or embedded systems is not an expectation at this level. Students might select an embedded device such as a car stereo, identify the types of data (radio station presets, volume level) and	

	procedures (increase volume, store/recall saved station, mute) it includes, and explain how the implementation details are hidden from the user.	
ICS-CS-02	**Compare levels of abstraction and interactions between application software, system software, and hardware layers.**	**Computing Systems** Hardware & Software
	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.	
ICS-CS-03	**Explain the roles of operating systems including memory management, data storage/retrieval, process management, and access control.**	**Computing Systems** Hardware & Software
	Examples of roles could include memory management, data storage/retrieval, processes management, and access control.	
ICS-CS-04	**Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.**	**Computing Systems** Troubleshooting
	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 adapt 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.	
ICS-NI-01	**Evaluate the relationship between routers, switches, servers, and topology with regard to networks.**	**Networks & the Internet** Network Communication & Organization
	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 destination. 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.	

ICS-NI-02	**Identify examples to illustrate how sensitive data can be affected by malware and other attacks.**	**Networks & the Internet** Cybersecurity
	Network security depends on a combination of hardware, software, and practices that control access to data and systems. The needs of users and the sensitivity of data determine the level of security implemented. 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.	
ICS-NI-03	**Recommend cybersecurity measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.**	**Networks & the Internet** Cybersecurity
	Security measures may include physical security tokens, two-factor authentication, and biometric verification. Potential security problems, such as denial-of-service attacks, ransomware, viruses, worms, spyware, and phishing, exemplify why sensitive data should be securely stored and transmitted. 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 to solve given problems or subproblems, such as through a cost-benefit analysis. Eventually, students should include more factors in their evaluations, such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns.	
ICS-NI-04	**Compare various security measures and consider tradeoffs between the usability and security of a computing system.**	**Networks & the Internet** Cybersecurity
	Security measures may include physical security tokens, two-factor authentication, and biometric verification, but 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.	
ICS-DA-01	**Compare different binary representations of data, including text, sound, images, and numbers.**	**Data & Analysis** Storage
	For example, convert hexadecimal color codes to decimal percentages, ASCII/Unicode representation, and logic gates.	
ICS-DA-02	**Evaluate the tradeoffs in how data elements are organized and where data is stored.**	**Data & Analysis** Storage
	People make choices about how data elements are organized and where data is stored. These choices affect cost, speed, reliability, accessibility, privacy, and integrity. Students should evaluate whether a chosen solution is most appropriate for a particular problem.	

	Students might consider the cost, speed, reliability, accessibility, privacy, and integrity tradeoffs between storing photo data on a mobile device versus in the cloud.	
ICS-DA-03	**Create interactive data visualizations using software tools to help others better understand real-world phenomena.**	**Data & Analysis** Collection Visualization & Transformation
	People transform, generalize, simplify, and present large data sets in different ways to influence how other people interpret and understand the underlying information. Examples include visualization, aggregation, rearrangement, and application of mathematical operations. 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 the system and evaluate these models against real-world observations. For example, flocking behaviors, queueing, or life cycles. Google Fusion Tables can provide access to data visualization online.	
ICS-DA-04	**Create computational models that represent the relationships among different elements of data collected.**	**Data & Analysis** Inference & Models
	Computational models make predictions about processes or phenomena based on selected data and features. 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. For example, students might create a simple producer–consumer ecosystem model using a programming tool. Eventually, they could progress to creating more complex and realistic interactions between species, such as predation, competition, or symbiosis, and evaluate the model based on data gathered from nature.	
ICS-AP-01	**Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.**	**Algorithms & Programming** Algorithms
	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 artifacts in response to a task or a computational problem that demonstrate the performance, reusability, and ease of implementation of an algorithm.	
ICS-AP-02	**Explain the use of artificial intelligence within computing systems.**	**Algorithms & Programming** Algorithms

	Artificial Intelligence (AI) has become a critical component of nearly all aspects of technology in our society. Students should be able to explain how AI is used within technology. For example, students might explain how a store makes recommendations based on search and buying history.	
ICS-AP-03	**Utilize lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.**	**Algorithms & Programming** 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.	
ICS-AP-04	**Justify the selection of specific control structures, considering implementation, readability, and program performance.**	**Algorithms & Programming** Control
	Implementation includes the choice of programming language, which affects the time and effort required to create a program. 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 and storage requirements; a quantitative analysis is not expected. Control structures at this level may include conditional statements, loops, event handlers, and recursion. For example, students might compare the readability and program performance of iterative and recursive implementations of procedures that calculate the Fibonacci sequence.	
ICS-AP-05	**Iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.**	**Algorithms & Programming** Control
	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. Students might create a mobile app that updates a list of nearby points of interest when the device detects that its location has been changed.	
ICS-AP-06	**Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.**	**Algorithms & Programming** Modularity
	At this level, students should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist. For example, students could create an app to solve a community problem by connecting to an online database through an application programming interface (API).	
ICS-AP-07	**Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.**	**Algorithms & Programming** Modularity

	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. Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. Modules allow for better management of complex tasks. The focus at this level is understanding a program as a system with relationships between modules. The choice of implementation, such as programming language or paradigm, may vary. Students could incorporate computer vision libraries to increase the capabilities of a robot or leverage open-source JavaScript libraries to expand the functionality of a web application.	
ICS-AP-08	**Systematically design programs for broad audiences.**	**Algorithms & Programming** Modularity
	Examples of programs could include games, utilities, and mobile applications. Students at lower levels collect feedback and revise programs. At this level, students should do so through a systematic process that includes feedback from broad audiences. Students might create a user satisfaction survey and brainstorm distribution methods that could yield feedback from a diverse audience, documenting the process they took to incorporate selected feedback in product revisions.	
ICS-AP-09	**Refine programs by incorporating feedback from users.**	**Algorithms & Programming** Modularity
	Examples of programs could include games, utilities, and mobile applications. Students at lower levels collect feedback and revise programs. At this level, students should do so through a systematic process that includes feedback from broad audiences. Students might create a user satisfaction survey and brainstorm distribution methods that could yield feedback from a diverse audience, documenting the process they took to incorporate selected feedback in product revisions.	
ICS-AP-10	**Evaluate licenses that limit or restrict use of computational artifacts when using resources such as software libraries.**	**Algorithms & Programming** Program Development
	Examples of software licenses include copyright, freeware, and the many open-source licensing schemes. At previous levels, students adhered to licensing schemes. At this level, they should consider licensing implications for their own work, especially when incorporating libraries and other resources. Students might consider two software libraries that address a similar need, justifying their choice based on the library that has the least restrictive license.	
ICS-AP-11	**Evaluate computational artifacts for usability.**	**Algorithms & Programming** Program Development

	Testing and refinement is the deliberate and iterative process of improving a computational artifact. Students should respond to the changing needs and expectations of end users and improve the performance, reliability, usability, and accessibility of artifacts. For example, students could solicit feedback from a variety of end users to help guide the size and placement of menus and buttons in a user interface.	
ICS-AP-12	**Modify computational artifacts to increase usability and accessibility.**	**Algorithms & Programming** Program Development
	The testing and refinement process of improving a computational artifact includes debugging (identifying and fixing errors) and comparing actual outcomes to intended outcomes. Students modify artifacts by incorporating feedback from a variety of end users to improve usability.	
ICS-AP-13	**Develop computational artifacts working in team roles using collaborative tools.**	**Algorithms & Programming** Program Development
	Collaborative tools could be as complex as a source code version control system or as simple as a collaborative word processor. Team roles in pair programming are driver and navigator but could be more specialized in larger teams. As programs grow more complex, the choice of resources that aid program development becomes increasingly important and should be made by the students. Students might work as a team to develop a mobile application that addresses a problem relevant to the school or community, selecting appropriate tools to establish and manage the project timeline; design, share, and revise graphical user interface elements; and track planned, in-progress, and completed components.	
ICS-AP-14	**Explain design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.**	**Algorithms & Programming** Program Development
	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. The development of complex programs is aided by resources such as libraries and tools to edit and manage parts of the program.	
ICS-IC-01	**Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.**	**Impacts of Computing** Culture
	Computing may improve, harm, or maintain 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. Students should also begin to identify potential bias during the design process to maximize accessibility in product design.	
ICS-IC-02	**Elaborate how computational innovations have and may continue to impact society.**	**Impacts of Computing** Culture

	Our world has been transformed by computational innovations. Students should be able to articulate how these innovations have impacted our society. For example, students might identify examples from education, healthcare, art/entertainment, or energy and create a presentation describing the impact.	
ICS-IC-03	**Evaluate how equity, access, and influence impact distribution of computing resources in a global society.**	**Impacts of Computing** Culture
	The distribution of computing resources is not uniform across all parts of our global society. Students should evaluate how equity and access are impacted by factors such as wealth, geographic location, governmental programs. For example, students might explore the effectiveness of the US E-rate program for increasing access to technology in schools.	
ICS-IC-04	**Test computational artifacts to reduce bias and equity deficits.**	**Impacts of Computing** Culture
	Biases could include incorrect assumptions developers have made about their user base. Equity deficits include minimal exposure to computing, access to education, and training opportunities. 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.	
ICS-IC-05	**Demonstrate ways a given algorithm applies to problems across disciplines.**	**Impacts of Computing** Culture
	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.	
ICS-IC-06	**Utilize tools and methods for collaboration on a project to increase connectivity of peers.**	**Impacts of Computing** Social Interactions
	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.	
ICS-IC-07	**Explain the beneficial and harmful effects that intellectual property laws can have on innovation.**	**Impacts of Computing** Safety Law & Ethics

	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, laws that mandate the blocking of some file-sharing websites may reduce online piracy but can restrict the right to access information. Firewalls can be used to block harmful viruses and malware but can also be used for media censorship. Students should be aware of intellectual property laws and be able to explain how they are used to protect the interests of innovators and how patent trolls abuse the laws for financial gain.	
ICS-IC-08	**Explain privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.**	**Impacts of Computing** Safety Law & Ethics
	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 can raise privacy concerns, such as social media sites mining an account even when the user is not online. Other examples include surveillance video used in a store to track customers for security or information about purchase habits or the monitoring of road traffic to change signals in real time to improve road efficiency without drivers being aware. Methods and devices for collecting data can differ by the amount of storage required, level of detail collected, and sampling rates.	
ICS-IC-09	**Evaluate the social and economic implications of privacy in the context of safety, law, and ethics.**	**Impacts of Computing** Safety Law & 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.	

High School Level 1 CS

HS-CS-01	**Describe the use of artificial intelligence within computing systems.**	**Computing Systems** Devices
	AI is present in nearly all computing systems. Students should be able to identify and describe how AI is playing a role in various industries and applications. Examples include digital ad delivery, self-driving cars, and credit card fraud detection.	
HS-CS-02	**Explain how computing devices manage and allocate shared resources.**	**Computing Systems** Hardware & Software
	Computing systems are often very complicated interconnected systems with numerous shared resources. Students need to understand how computing systems manage these resources and avoid conflicts.	
HS-CS-03	**Illustrate the ways computing systems implement logic, input, and output through hardware components.**	**Computing Systems** Troubleshooting
	While much of what students perceive as computer science revolves around the input and output of computing systems which on the surface appears to be purely software, students should understand how every piece of software relies on the physical hardware that translates the code into a computing solution. Students should explore the hardware implementations of such components as logic gates and IO pins.	
HS-CS-04	**Utilize guidelines that convey systematic troubleshooting strategies that debug computer systems.**	**Computing Systems** Troubleshooting
	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 adapt 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.	
HS-NI-01	**Identify issues of network functionality in computational artifact design.**	**Networks & the Internet** Network Communication & Organization
	As more computational artifacts rely on online resources or components for functionality, students should be able to identify issues that may arise from connected applications (bandwidth, latency, security). There are online simulators that can help students explore these issues.	

HS-NI-02	**Analyze issues of network functionality in computational artifact design.**	**Networks & the Internet** Network Communication & Organization
	As students work with network dependent artifacts, they should be able to test and analyze the impacts of networking issues on artifacts. Students can test artifacts on various devices and utilize different networks.	
HS-NI-03	**Identify issues of unauthorized access and cybersecurity in computational artifact design.**	**Networks & the Internet** Cybersecurity
	Among the most challenging aspects of network connected artifacts is the inherent threat of unauthorized access and cybersecurity issues. Students should be able to identify the risks posed by these threats.	
HS-NI-04	**Analyze issues of unauthorized access and cybersecurity in computational artifact design.**	**Networks & the Internet** Cybersecurity
	Students should be able to recognize particular security threats and compare ways developers might protect against them. In particular, students should explore examples of mitigation strategies such as encryption and authentication strategies, secure coding, and safeguarding keys.	
HS-NI-05	**Explain tradeoffs when selecting and implementing cybersecurity recommendations for various scenarios based on factors such as efficiency,**	**Networks & the Internet** Cybersecurity
	Network security depends on a combination of hardware, software, and practices that control access to data and systems. The needs of users and the sensitivity of data determine the level of security implemented. Every security measure involves tradeoffs between the accessibility and security of the system. Students should be able to describe, justify, and document choices they make using terminology appropriate for the intended audience and purpose. Students could debate issues from the perspective of diverse audiences, including individuals, corporations, privacy advocates, security experts, and government.	
HS-DA-01	**Identify patterns in data representing complex systems with select data analysis tools and techniques.**	**Data & Analysis** Collection Visualization & Transformation
	Students should be able to identify patterns in data representing complex systems, such as identify trends in a dataset representing social media interactions, movie reviews, or shopping patterns.	

HS-DA-02	**Select appropriate data collection tools and techniques.**	**Data & Analysis** Collection Visualization & Transformation
	The use of data in computing artifacts is one of the dominant forces shaping technology today. Students should be familiar with data collection tools (surveys, scientific probes, online repositories) and techniques (harvesting, crowd-sourcing, simulations). Students should be able to select the appropriate tools and techniques for a given task.	
HS-DA-03	**Compile data sets that support a claim or communicate information.**	**Data & Analysis** Collection Visualization & Transformation
	Communicating with data is a critical component for many technology innovations. Whether it is rank ordering possible purchases on an ecommerce site or displaying the progress in a video game, data is often used to communicate information to the end user. Students should be able to articulate a story or impact through data and visualizations.	
HS-DA-04	**Identify the ability of models and simulations to test hypotheses.**	**Data & Analysis** Inference & Models
	Especially in scientific research, computers are playing an increasing role in testing hypotheses. Because computers have the ability to run millions of simulations in the time a lab experiment might conduct a handful, modeling and simulations are critical to modern science. Students should explore the application of modeling to solve society's challenges, recent examples include medicine, industrial design, and environmental science.	
HS-DA-05	**Formulate hypotheses with select models and simulations.**	**Data & Analysis** Inference & Models
	Students should experience models and simulations in hands-on experiences. They should use existing models and simulations to formulate hypotheses. These can be online interactives (Phet) or could be specific applications written for the class to test.	
HS-AP-01	**Identify artificial intelligence algorithms.**	**Algorithms & Programming** Algorithms
	As algorithms are a foundational component of computer science, students should recognize the role of algorithms in machine learning. Among the most common algorithms are regression, decision trees, and search. By exploring these algorithms, students will begin to develop a foundation for implementing AI.	

HS-AP-02	**Solve computational problems with classic algorithms.**	**Algorithms & Programming** Algorithms
	Algorithms are a foundational component of computer science. There are numerous foundational algorithms that serve as the basis for many applications. Students should be familiar with and should implement many of the classic algorithms, especially those for sorting (bubble, selection) and search (binary, linear) .	
HS-AP-03	**Evaluate algorithms in terms of their efficiency, correctness, and clarity.**	**Algorithms & Programming** Algorithms
	When selecting appropriate algorithms, students must make decisions based on the resources available and the constraints of the problem. Students should evaluate various algorithms for efficiency, correctness, and clarity when implementing. Search and sort algorithms are ideal for such explorations.	
HS-AP-04	**Select an appropriate data structure for information of a given problem.**	**Algorithms & Programming** Variables
	Managing data within a computational artifact is critical. Students should be able to implement data storage using various methods. In addition to understanding primitive data types (int, float, char, string), students should be able to implement data structures such as lists, arrays, stacks, and queues.	
HS-AP-05	**Illustrate the flow of execution of a recursive algorithm.**	**Algorithms & Programming** Control
	While recursive algorithms are relatively simplistic in design, they can be quite challenging to implement and debug. Students should be able to trace the implementation of recursive algorithms. Recursive sorting and searching are good algorithms to explore.	
HS-AP-06	**Identify a large-scale computational problem.**	**Algorithms & Programming** Modularity
	Students should be able to identify real-world problems that span several domains which represent opportunities for large-scale computational solutions. Students might explore the Grand Challenges organization for ideas.	
HS-AP-07	**Analyze general patterns applicable to a solution.**	**Algorithms & Programming** Modularity

	As students encounter complex, real-world problems that span multiple disciplines or social systems, they should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist. For example, students could create an app to solve a community problem by connecting to an online database through an application programming interface (API).	
HS-AP-08	**Create computational artifacts with pre-existing procedures, external components, libraries and APIs.**	**Algorithms & Programming** Modularity
	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. Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. Modules allow for better management of complex tasks. The focus at this level is understanding a program as a system with relationships between modules. The choice of implementation, such as programming language or paradigm, may vary. Students could incorporate computer vision libraries to increase the capabilities of a robot or leverage open-source JavaScript libraries to expand the functionality of a web application.	
HS-AP-09	**Create a computational artifact through an industry-standard process.**	**Algorithms & Programming** Program Development
	As much as computer science involves knowledge and skills withinin program development, the real application of computer science also requires an understanding of the key processes used in the workplace. Students should begin to develop comfort with these processes, including agile, spiral, or waterfall.	
HS-AP-10	**Justify that a computational artifact meets design specifications with systematic testing and debugging methods.**	**Algorithms & Programming** Program Development
	Students should develop and use a series of test cases to verify that a program performs according to its design specifications.	
HS-AP-11	**Construct a computational artifact as a team through industry appropriate collaborative tools and processes.**	**Algorithms & Programming** Program Development
	When developing software, development teams must use tools designed to facilitate collaboration and to manage version control. Students should work in teams to experience these challenges and to understand industry solutions, such as GITHUB.	
HS-AP-12	**Compose standard documentation for computational artifacts to make it easier to follow, test, and debug.**	**Algorithms & Programming** Program Development

	As part of best practices, especially when working on development teams, students should implement appropriate documentation strategies. These strategies make it easier for teammates to better follow the logic and debug any issues.	
HS-AP-13	**Modify an existing computational artifact for additional functionality.**	**Algorithms & Programming** Program Development
	While students should be able to develop computational artifacts of their own design, it is also critical that students be able to work on existing artifacts. Students should be comfortable with taking an existing artifact and making enhancements or improvements.	
HS-AP-14	**Discuss intended and unintended implications of a modified computational artifact.**	**Algorithms & Programming** Program Development
	As students modify an existing application, they must be aware that all changes have implications, some intended and some not. Students should be prepared to justify these tradeoffs and to document the implications. For instance, changes made to a method or function signature could break invocations of that method elsewhere in a system.	
HS-AP-15	**Develop computational artifacts for multiple platforms.**	**Algorithms & Programming** Program Development
	As students work to create computational artifacts, they should be aware that end users may experience the artifact on various devices. As such, students should design and develop their solution with various experiences and operating systems in mind. Students might develop multi-platform solutions that work across computer desktop, web, and mobile.	
HS-IC-01	**Evaluate computational artifacts for their effects on society.**	**Impacts of Computing** Culture
	As much as humans are shaping the technology innovations in society, it is also the case that technology is shaping society. Students should be able to recognize how recent technological innovations have impacted our world. Students may consider the Internet broadly, but should also examine specific applications on the web, including online e-commerce, social networking, and collaboration tools.	
HS-IC-02	**Make computational artifact recommendations for maximized beneficial and minimal harmful effects on society.**	**Impacts of Computing** Culture
	Everyone who leverages the power of computing to create innovative products bears the responsibility for minimizing the harmful impacts that their work has on society. Students should use existing innovations to provide meaningful feedback on how those artifacts might maximize their beneficial effects and minimize harmful effects on society.	
HS-IC-03	**Predict how computational innovations that revolutionized aspects of our culture might evolve.**	**Impacts of Computing** Culture

	Technology is not static. Our use of digital tools will only increase with time. Students might consider how these technologies might evolve especially in the areas of education, healthcare, art/entertainment, and energy.	
HS-IC-04	**Evaluate how equity, access, and influence impact distribution of computing resources in a global society.**	**Impacts of Computing** Culture
	The distribution of computing resources is not uniform across all parts of our global society. Students should evaluate how equity and access are impacted by factors such as wealth, geographic location, governmental programs. For example, students might explore the effectiveness of the US E-rate program for increasing access to technology in schools.	
HS-IC-05	**Create computational artifacts to ensure accessibility and reduce computational bias.**	**Impacts of Computing** Culture
	Biases could include incorrect assumptions developers have made about their user base. Equity deficits include minimal exposure to computing, access to education, and training opportunities. 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.	
HS-IC-06	**Utilize tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields.**	**Impacts of Computing** Social Interactions
	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.	