Skill Struck’s alignment to

Utah 6-12 Computer Science Standards Fall 2019

Legend

✅ = Standard aligned

🔹 = Not currently aligned

Standard	Status
6.CS.1 Utilize troubleshooting strategies to resolve hardware and software issues in a logical order. (Practice 4: Developing and Using Abstractions) Students will be able to utilize a step-by-step approach to identify and resolve problems with hardware and software. For example, a checklist can be used to ensure that possible solutions are not overlooked such as checking for writing conventions before finalizing a writing assignment. Students may refer to the order of operations when solving a math equation. Students may search for technical information online when solving problems. A flow diagram may be used to determine possible next steps.	✅
6.NI.1 Explain potential security threats and practice protective measures to reduce these threats. (Practice 4: Developing and Using Abstractions) Students will recognize and explain the existence of threats and protect their personal information using appropriate security measures. Students identify multiple methods for protecting their data and articulate the value	✅

and appropriateness for each method. For example, students should develop habits such as logging off devices and maintaining hidden, strong, evolving passphrases. Also, understanding how to use cybersecurity to protect personal and sensitive data.	
6.DA.1 Represent a single data set in multiple ways using words, symbols, manipulatives, charts, diagrams, and visuals. (Practice 4: Developing and Using Abstractions.) Students will represent data in multiple ways using abstraction. For example, convert letters into binary code, location into GPS coordinates or ideas and phrases into emojis. Students may represent a location as a string "New Zealand" or a numeric input (longitude/latitude geolocation). Another example could be representing colors using binary, hexadecimal, or words.	✅
6.AP.1 Design and illustrate algorithms to efficiently solve complex problems by utilizing pseudocode and/or other descriptive methods. (Practice 3: Recognizing and defining computational problems) Students will decompose or design algorithms (how to instructions) utilizing pseudocode to solve complex problems. Students will be able to decompose a real-world problem and illustrate the decision-making process in a well-organized flowchart, storyboard, ordered directions, notations, or other method. For example, the students might create a flowchart to illustrate which equipment to use for recess based on the weather, play preference, and a student's energy level.	✅
6.AP.2 Create naming conventions for variables that support the debugging process and incorporate these variables into a simple program. (Practice 7: Communicating about Computing) To make the debugging process easier, students will create and name variables that store data in a meaningful and logical way. For example, when writing an algorithm, students will incorporate names based on the command function such as use the variable "turn" to describe direction, "loop" for repeating tasks.	✅

6.AP.3 Annotate programs in order to document their use and improve readability, testing, and debugging. (Practice 7: Communicating about computing) Students will annotate by adding descriptors, comments or notations to describe a program for future use and easier debugging. For example, students could add comments to describe the functionality of different segments of code. These annotations are like those in textbooks and instruction manuals or note-taking on a presentation slide.	✅
6.IC.1 Recognize and discuss issues of bias and accessibility in existing technologies. (Practice 1: Fostering an inclusive computer culture. Practice 7: Communicating about computing.) Students will be able to recognize and discuss the usability and accessibility of various technology tools such as apps, games, and devices acknowledging designer bias. For example, students could discuss if devices in their school are ADA compliant and whether software they use has been designed for a particular user or a diverse population.	✅
7.CS.1 Design modifications to computing devices in order to improve the ways users interact with the devices. (Practice 3: Recognizing and Defining Computational Problems.) Students will be able to identify problems with existing computing devices or technologies and design modifications to improve the ways users interact with those technologies. For example, students may design changes to an existing device in order to improve accessibility for users with visual, audio, physical, language and/or other barriers or students may redesign an existing computing device to be more functional for an everyday user.	✅
7.NI.1 Model the role of protocols in transmitting data across networks and the Internet. (Practice 7: Communication about Computing) Students will model how protocols such as HTTP and TCP/IP allow for the transmission of data across networks and the internet. For example, students will participate in a role play and physically act out the data	✅

transmission process following protocols (a set of rules).	
7.DA.1 Collect data using computational tools and transform the data to make it more useful. (Practice 2: Collaborating about Computing.) Students will use computational tools to collect and transform data in a real-world scenario or applications. For example, students will use a Microbit circuit board to collect temperatures, soil moisture levels, etc. and use a program/app to create a data visualization. Additionally, students may create and administer a survey in a social studies class to aggregate data on a pertinent topic and then create a chart or graph to better display the data.	✅
7.AP.1 Design and iteratively develop programs that combine control structures. (Practice 5: Creating Computational Artifacts; Practice 6: Testing and Refining Computational Artifacts) Students will design, develop, test, and refine programs using control structures such as loops or conditional logic statements. For example, students will create a choose your own adventure story/presentation, a flowchart, or code a simple interactive game or animation.	✅
7.AP.2 Seek and incorporate feedback from team members and users to refine a solution to a programming project that meets the user's needs. (Practice 2: Collaborating Around Computing; Practice 6: Testing and Refining Computational Artifacts.) Students will collaborate to seek and incorporate feedback from team members on a team project and use that feedback to refine their project to meet the needs of all users. For example, students will solicit feedback from others on a programming project to improve the quality of their work and meet the needs of all users.	✅
7.AP.3 Systematically test and refine programs using a range of test cases. (Practice 6: Testing and Refining Computational Artifacts.) Students will use a variety of problem-solving processes such as the engineering design process, decision matrix, pros and cons, or DMAIC (define, measure, analyze, improve and control) to test and refine a	✅

project or program. Students will test and refine a computer program, an engineering artifact, or solution. For example, students may test and refine a math program solving for surface area of different shapes (triangles, quadrilaterals, polygons, cubes).	
7.AP.4 Select and assign tasks to maintain a project timeline when collaboratively developing computational artifacts. (Practice 2: Collaborating Around Computing. Practice 5: Creating Computational Artifacts.) Students will select, assign, and manage tasks within a project timeline of milestones and due dates while collaboratively working on projects. For example, students will use tools such as storyboards, to-do lists, team roles, and other project management tools to organize their projects and share the work across team members and help them be more efficient in managing time and resources.	✅
7.IC.1 Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options. (Practice 1: Fostering an Inclusive Computing Culture; Practice 7: Communicating about Computing.) Advancements in computer technology have trade-offs. Students will 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	✅
8.CS.1 Design a project that combines hardware and software components to collect and exchange data. (Practice 5: Creating Computational Artifacts; Practice 4: Developing and Using Abstractions) Students will use hardware (computer, tablet, mobile device, etc.) and appropriate software (word processing, presentation, spreadsheet, movie maker/editing, etc.) to design a project. For example, students can create	✅

a news broadcast related to the Great Depression. For example, students can collect information (interview) of how the depression affected each group of people, the economic impacts, and how the depression impacted Utah's economy.	
8.NI.1 Explain how proper protocols transmit data across networks and the internet. (Practice 4. Developing and Using Abstractions) Students will understand rules are developed to deliver data that is broken down into packets (smaller bits of data) to travel across networks and the internet. Students will explain data is delivered in a fast and secure path to avoid missing information. For example, students can create a plan of action to deliver supplies needed in a national disaster. They will need to determine the best route(s) for quick and secure delivery of supplies.	✅
8.DA.3 Test and analyze the effects of changing variables in models/simulations. (Practice 3. Recognizing and Defining Computational Problems; Practice 4. Developing and Using Abstractions; Practice 5. Creating Computational Artifacts) Students will demonstrate how changing variables will affect outcomes in a model/simulation. For example, students will understand the relationship between the mass and speed of objects and the relative amount of kinetic energy of the objects. Students can test and analyze a full cart vs. an empty cart or rolling spheres with different masses down a ramp to measure the effects on stationary masses.	✅
8.AP.1 Develop a program with iterative protocols that combine control structures and use compound conditions. (Practice 5. Creating Computational Artifacts; Practice 6. Testing and Refining Computational Artifacts) Students will develop programs that use compound conditions (True/False, If/Then, etc.) and loops. The development process should include multiple phases and pseudocode. For example, students will understand the relationship of cause and effect relationships in particle motion, temperature, density, and the state of a pure substance when	✅

heat energy is added or removed. Students can create true/false and if/then statements in the development process showing the results of adding and removing heat energy and the cause and effect it has on different substance's states.	
8.AP.2 Create procedures with or without parameters to organize code and make it easier to reuse. (Practice 4. Developing and Using Abstractions; Practice 5. Creating Computational Artifacts) Students will organize code that can be reused with or without parameters. Students will create procedures that can identify properties of functions. Students will be able to demonstrate the properties of two functions based on x and y values.	✅
8.AP.3 Create a new program incorporating existing code, media, and libraries; and give proper attribution. (Practice 2. Collaborating Around Computing; Practice 4. Developing and Using Abstractions; Practice 5. Creating Computational Artifacts; Practice 7. Communicating about computing) Students will write original programs that incorporate someone else's code and/or media and give proper attribution to the source. Students can manipulate an existing file from a block code program (i.e. Scratch) to demonstrate the conflicts during the American expansion as American Indians were forced from their lands and the tensions over slavery.	✅
9/10.CS.1 Describe ways in which the specific implementation details of a computing system are hidden by abstractions in order to manage complexity. (Practice 4. Developing and Using Abstractions; Practice 7. Communicating About Computing) Students will describe how layers of generality simplify the users experience by hiding many of the complex details. For example, the summation symbol Σ indicates that you are adding all terms instead of writing each term individually with plus signs in between. Students could also explain the challenges Alan Turing faced and the process he used in breaking Enigma.	✅

9/10.CS.2 Identify the different levels of abstraction in a computer system. (Practice 4. Developing and Using Abstractions; Practice 7: Communicating About Computing) Students will identify different layers of computing abstraction which could include applications, operating systems, and hardware. For example, an educational app (ex. CANVAS) utilizes the phone's hardware and communicates to the "app" on the phone to send assignments to teachers or comment on a discussion board. Another example of this is describing the functions of the different systems of the body and how they work together to make the body function.	✅
9/10.CS.3 Develop guidelines that communicate systematic troubleshooting strategies that others can use to identify and fix errors. (Practice 6. Testing and Refining Computational Artifacts.) Students will develop strategies for troubleshooting and fixing problems and/or errors in a system. 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.7 For example, students will design a solution to a space exploration challenge by breaking it down into smaller, more manageable problems that can be solved through the structure and function of a device. Define the problem, identify criteria and constraints, develop possible solutions using models, analyze data to make improvements from iteratively testing solutions, and optimize a solution. Examples of problems could include, cosmic radiation exposure, transportation on other planets or moons, or supplying energy to space travelers.	✅
9/10.NI.1 Describe essential elements for connecting to a network and identify issues that impact network functionality. (Practice 7: Communicating About Computing.) Students will describe which hardware, software, and information are	✅

needed to connect to the internet. Students will also identify issues that might slow down a network connection (overloaded cell phone towers, sporting events, and natural disasters). Teachers may utilize an online network simulator to demonstrate network functionality.	
9/10.NI.2 Describe the design structure of the internet and identify standard protocols. (Practice 4: Developing and Using Abstractions) Students will describe how the internet is designed to have multiple paths to any two things that are connected, in case one path is compromised. They will also describe how standard rules allow everything to connect to one network. For example, students can discuss how they can drive home using a different path if the road is closed on one possible path. They can also discuss how the traffic rules help them travel safely.	✅
9/10.DA.1 Demonstrate different representations of data (numbers, characters, and images). (Practice 4: Developing and Using Abstractions) Students will be able to represent data or information in different forms. For example, students will decipher a message in binary code using an alphanumeric key. Students will understand that images or logos could be used to portray information as well.	✅
9/10.DA.2 Describe disadvantages or benefits associated with how data elements are organized and stored. (Practice 3. Recognizing and Defining Computational Problems; Practice 7: Communicating About Computing) Students will describe the properties for a given data set or proper storage choice considering a specific problem [file types, compression (Lossy vs. Lossless), speed, file size, accessibility]. For example, students will determine the best option for storing photos or music, whether it be on mobile vs computer vs cloud and describe the benefits or costs associated with each method.	✅
9/10.DA.3 Create data visualizations to help others better understand real-world phenomena or factual data information. (Practice 5. Creating Computational Artifacts; Practice 7: Communicating About Computing; Practice 4. Developing and Using Abstractions)	

Students will create data visualizations using factual data to better interpret the information. For example, students could develop a chart marking the stock market trends and pertinent historic events (either societal or technological events) to see what types of events affect the stock market in a negative or positive manner.	
9/10.AP.1 Design algorithms to solve computational problems using a combination of original and existing algorithms (Practice 3. Recognizing and Defining Computational Problems; Practice 4: Developing and Using Abstractions) Students will create algorithms that combine existing algorithms with their original program to complete a certain task. For example, students could use the formula for energy of motion to construct a device that converts one form of energy into another form of energy to solve a complex real-life problem.	✅
9.10.AP.2 Create more generalized computational solutions using collections of items (like an array or list) instead of separating using individual items. (Practice 4: Developing and Using Abstractions) Students will create groups of items using sorting methods by grouping like items together to refer to all at once. For example, students could chart the number of immigrants by nationality that entered the United States during the beginning of the industrial age.	✅
9/10.AP.3 Decompose problems into multiple smaller problems through systematic analysis, using constructs (such as procedures, modules, functions, methods, and/or classes). (Practice 3. Recognizing and Defining Computational Problems) Students will break down a big or complex problem and split it into smaller, easier-to-manage components. For example, students will find roots of polynomials by factoring them into smaller components and then solving for each factor.	✅
9/10.AP.4 Create computational artifacts using modular design. (Practice 5: Creating Computational Artifacts)	✅

Students will create a computational artifact to solve a complex problem by breaking down the problems into smaller, easier-to-manage components. For example, students can solve a complex math problem using the order of operations.	
9/10.AP.5 Identify and collaboratively suggest changes to an application's design using feedback from a variety of users. (Practice 7: Communicating About Computing) Students will identify that when they are designing a program or product for a client, they must listen to the clients' needs and wants as well as be willing to accept feedback from peers. For example, students will create or redesign a company logo for a certain company, conduct focus group research on their design, and make proper design corrections based on the feedback.	✅
9/10.AP.6 Explain the limitations of licenses that restrict computational artifacts when using resources created by others. (Practice 7: Communicating About Computing) Students will demonstrate knowledge of different copyright licenses for software use and when to give proper reference. For example, students can research different types of patents and copyright laws that were established during the industrial age and compare them to the intellectual property laws of modern- day patents and licenses.	✅
9/10.AP.7 Iteratively evaluate and refine a computational artifact to enhance its performance, reliability, usability, and accessibility. (Practice 6: Testing and Refining Computational Artifacts) Students will evaluate how computational artifacts can be developed, tested, and edited repeatedly to improve performance, ease of use, reliability, and/or accessibility. For example, students will use the scientific method to design an air powered rocket to land hit a target from a specific distance. This could also be a great opportunity to introduce Moore's Law.	✅
9/10.AP.8 Design and develop computational artifacts using collaborative	✅

tools. (Practice 2: Collaborating Around Computing; Practice 7: Communicating About Computing) Students will use collaborative tools to design and develop computational artifacts as a team. For example, students can collaborate on a presentation using cloud-based applications (Office 365, Google suite, etc.) to complete the design and development process of a project.	
9/10.AP.9 Create documentation (pseudocode) that communicates the design of the solution to a complex problem using text, graphics, and/or demonstrations. (Practice 7: Communicating About Computing) Students will design solutions to problems and document these solutions—using pseudocode, flowcharts, and other means--so that they can be implemented by either the student or someone else. During and after implementation, comments and additional documentation can facilitate future maintenance of that process. For example, students will create an outline for an essay before starting on the rough draft.	✅
9/10.IC.1 Evaluate how computing has impacted and/or impacts personal, ethical, social, economic, and cultural practices. (Practice 3. Recognizing and Defining Computational Problems; Practice 7: Communicating About Computing) Students will determine how computing has positively and/or negatively impacted the world around us. For example, students can research the impact computing has had on society, and as a class, be put into affirmative and negative teams to debate the effects of computing.	✅
9/10.IC.2 Understand that bias is always introduced when creating computational artifacts, identify ways that this unintended bias may impact equity, and then evaluate methods for alleviating that impact. (Practice 1: Fostering an Inclusive Computing Culture) Students will understand that bias may impact their work and devise solutions for overcoming that bias. When creating computational artifacts, such as software applications, the programmer's experience, culture, values, and knowledge influences the design and outcome. This may	✅

inadvertently discriminate against specific groups of users. For example, students can team up to describe how ethnicity affects facial recognition and speech to text functionality in technology, and how to resolve those issues.	
9/10.IC.3 Identify solutions to problems in other content areas using established algorithms. (Practice 1: Fostering an Inclusive Computing Culture; Practice 2: Collaborating Around Computing) Students will develop solutions to problems that can relate to other subject areas. They will create and analyze a step-by-step process and apply it to a problem relevant to cross-curricular subjects. For example, students can examine the steps involved in solving a quadratic equation.	✅
11/12.NI.1 Identify types of security threats, and then compare and contrast measures that can be used to address, resolve, and/or prevent identified threats. (Practice 3. Recognizing and Defining Computational Problems; Practice 7: Communicating About Computing) Students will identify and evaluate different types of security threats and determine potential solutions with justification. For example, students will role play or act out different security threats, in a group, while also showing how to combat that security threat.	✅
11/12.NI.2 Compare and contrast cryptographic techniques to model the secure transmission of information (data). (Practice 3. Recognizing and Defining Computational Problems; Practice 5. Creating Computational Artifacts; Practice 7: Communicating About Computing) Students will demonstrate an understanding of how information is transformed/manipulated via cryptography by creating an encryption algorithm. For example, students will understand how Alan Turing was able to break the Enigma code in World War II. Students will then create their own cypher and share among their peers.	✅
11/12.DA.1 Refine or create computational artifacts to better represent the relationships among different elements of data collected from factual sources or other processes. (Practice 3. Recognizing and Defining	✅

Computational Problems; Practice 4: Developing and Using Abstractions; Practice 5: Creating Computational Artifacts; Practice 6: Testing and Refining Computational Artifacts; Practice 7: Communicating About Computing) Students create and refine a computational model of data to explain the relationships between the different components of the model. For example, students will write a persuasive essay comparing the Allies and Axis of Power of World War II.	
11/12.AP.1 Iteratively design and develop computational artifacts for practical, personal, or societal expression that implements an algorithm based on the result of an evaluation or user input. (Practice 2: Collaborating Around Computing Practice 3: Recognizing and Defining Computational Problems; Practice 5: Creating Computational Artifacts; Practice 6: Testing and Refining Computational Artifacts) Students design and create a computational artifact that develops and implements algorithms (steps) based on the results of an evaluation of a result or user input. For example, students can brainstorm ideas for creating solutions to energy problems with prioritized criteria and trade-offs while considering cost, safety, reliability, as well as possible social, cultural, and environmental impacts.	✅
11.12.AP.2 Systematically design and create programs for broad audiences by incorporating feedback from users. (Practice 1: Fostering an Inclusive Computing Culture; Practice 2: Collaborating Around Computing; Practice 3. Recognizing and Defining Computational Problems; Practice 4: Developing and Using Abstractions; Practice 5: Creating Computational Artifacts; Practice 6: Testing and Refining Computational Artifacts; Practice 7: Communicating About Computing) Students will review and evaluate feedback from users and then redesign a program (process) to reflect identified needs from user data. For example, students will create a marketing advertisement for a certain product, conduct focus group research on their advertising design, and make proper design corrections based on the feedback.	✅

11/12.AP.3 Design and develop computational artifacts working in team roles using collaborative tools. (Practice 2: Collaborating Around Computing; Practice 4: Developing and Using Abstractions; Practice 5: Creating Computational Artifacts; Practice 6: Testing and Refining Computational Artifacts; Practice 7: Communicating About Computing) Students will collaborate to design and develop multiple artifacts in teams. For example, students will work together to develop a video game in their subject matter expert roles, which may include, writer, programmer, artist, audio, etc.	✅
11/12.AP.4 Produce documentation to support the decisions made during the design and creation process using text, graphics, presentations, and/or demonstrations in the development of complex programs. (Practice 3. Recognizing and Defining Computational Problems; Practice 4: Developing and Using Abstractions; Practice 5: Creating Computational Artifacts; Practice 6: Testing and Refining Computational Artifacts; Practice 7: Communicating About Computing) Students will produce documented decisions made during the design and creation process using text, graphics, presentations, and demonstrations in the development of complex programs. For example, students will create instructions to use building blocks (like LEGO's) to instruct others to recreate their design.	✅
11/12.IC.1 Evaluate and discuss the ways computing impacts personal, ethical, social, economic, and cultural practices. (Practice 1: Fostering an Inclusive Computing Culture; Practice 2: Collaborating Around Computing; Practice 3. Recognizing and Defining Computational Problems; Practice 7: Communicating About Computing) Students will evaluate and discuss the ways computing impacts personal, ethical, social, economic, and cultural practices. For example, students will research a current event that is relevant to computer science, take a side (either pro or con), and debate their findings in class.	✅
11/12.IC.2 Identify impacts of bias and equity deficits on design and	✅

implementation of computational artifacts, while evaluating appropriate processes for identifying issues of bias. (Practice 1: Fostering an Inclusive Computing Culture; Practice 2: Collaborating Around Computing; Practice 3. Recognizing and Defining Computational Problems; Practice 5: Creating Computational Artifacts; Practice 6: Testing and Refining Computational Artifacts; Practice 7: Communicating About Computing) Students will understand that bias may impact their work and devise solutions for overcoming that bias. When creating computational artifacts, such as software applications, the programmer's experience, culture, values, and knowledge influences the design and outcome. This may inadvertently discriminate against specific groups of users. For example, students can describe how a self-driving car can decide what action to take when every possible action leads to an accident--the programmer must account for these possibilities and the values and culture of the programmer will inform this decision.	
11/12.IC.3 Demonstrate computational thinking using algorithms to problem solving across multiple disciplines. (Practice 3. Recognizing and Defining Computational Problems; Practice 4: Developing and Using Abstractions; Practice 6: Testing and Refining Computational Artifacts; Practice 7: Communicating About Computing) Students will demonstrate ways to problem-solve across disciplines. For example, students can use computational thinking and patterns to predict certain genetic traits in chromosomes that will be passed on from parents to offspring.	✅