| | | | | | |
|---|---|---|---|---|---|
| 6-8 | 6-8.CS.1 | Design modifications to computing devices in order to improve the ways users interact with the devices. | Computing devices can extend the abilities of humans, but design considerations are critical to make these devices useful. Students suggest modifications to the design of computing devices and describe how these modifications would improve usabilty. | | |
| | | | | | |
| For example, students could create a design for the screen layout of a smartphone that is more usable by people with vision impairments or hand tremors. They might also design how to use the device as a scanner to convert text to speech. | | | | | |
| | | | | | |
| Alternatively, students could design modifications for a student ID card reader to increase usability by planning for scanner height, need of scanner device to be connected physically to the computer, robustness of scanner housing, and choice of use of RFID or line of sight scanners. (CA NGSS: MS-ETS1-1) | Computing Systems | Devices | Inclusion, Computational Problems | 1.2, 3.3 | |
| 6-8 | 6-8.CS.2 | Design a project that combines hardware and software components to collect and exchange data. | Collecting and exchanging data involves input, output, storage, and processing. When possible, students select the components for their project designs by considering tradeoffs between factors such as functionality, cost, size, speed, accessibility, and aesthetics. Students do not need to implement their project design in order to meet this standard. | | |
| | | | | | |
| For example, students could design a mobile tour app that displays information relevant to specific locations when the device is nearby or when the user selects a virtual stop on the tour. They select appropriate components, such as GPS or cellular-based geolocation tools, textual input, and speech recognition, to use in their project design. | | | | | |
| | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Alternatively, students could design a project that uses a sensor to collect the salinity, moisture, and temperature of soil. They may select a sensor that connects wirelessly through a Bluetooth connection because it supports greater mobility, or they could instead select a physical USB connection that does not require a separate power source. (CA NGSS: MS-ETS1-1, MS-ETS1-2) | Computing Systems | Hardware & Software | Creating | 5.1 | |
| 6-8 | 6-8.CS.3 | Systematically apply troubleshooting strategies to identify and resolve hardware and software problems in computing systems. | When problems occur within computing systems, it is important to take a structured, step-by-step approach to effectively solve the problem 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. 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. | | |
| | | | | | |
| For example, students could work through a checklist of solutions for connectivity problems in a lab of computers connected wirelessly or through physical cables. They could also search for technical information online and engage in technical reading to create troubleshooting documents that they then apply. (CA CCSS for ELA/Literacy RST.6-8.10) | | | | | |
| | | | | | |
| Alternatively, students could explore and utilize operating system tools to reset a computer's default language to English. | | | | | |
| | | | | | |
| Additionally, students could swap out an externally-controlled sensor giving fluctuating readings with a new sensor to check whether there is a hardware problem. | Computing Systems | Troubleshooting | Testing | 6.2 | |

| | | | | | |
|---|---|---|---|---|---|
| 6-8 | 6-8.NI.4 | Model the role of protocols in transmitting data across networks and the Internet. | Protocols are rules that define how messages between computers are sent. They determine how quickly and securely information is transmitted across networks, as well as how to handle errors in transmission. Students model how data is sent using protocols to choose the fastest path and to deal with missing information. Knowledge of the details of how specific protocols work is not expected. The priority at this grade level is understanding the purpose of protocols and how they enable efficient and errorless communication. | | |
| | | | | | |
| For example, students could devise a plan for sending data representing a textual message and devise a plan for resending lost information. | | | | | |
| | | | | | |
| Alternatively, students could devise a plan for sending data to represent a picture, and devise a plan for interpreting the image when pieces of the data are missing. | | | | | |
| | | | | | |
| Additionally, students could model the speed of sending messages by Bluetooth, Wi-Fi, or cellular networks and describe ways errors in data transmission can be detected and dealt with. | Networks & the Internet | Network Communication & Organization | Abstraction | 4.4 | |
| 6-8 | 6-8.NI.5 | Explain potential security threats and security measures to mitigate threats. | Cybersecurity is an important field of study and it is valuable for students to understand the need for protecting sensitive data. Students identify multiple methods for protecting data and articulate the value and appropriateness for each method. Students are not expected to implement or explain the implementation of such technologies. | | |
| | | | | | |
| For example, students could explain the importance of keeping passwords hidden, setting secure router administrator passwords, erasing a storage device before it is reused, and using firewalls to restrict access to private networks. | | | | | |
| | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Alternatively, students could explain the importance of two-factor authentication and HTTPS connections to ensure secure data transmission. | Networks & the Internet | Cybersecurity | Computational Problems | 3.1, 3.3 | |
| 6-8 | 6-8.NI.6 | Apply multiple methods of information protection to model the secure transmission of information. | Digital information is protected using a variety of cryptographic techniques. Cryptography is essential to many models of cybersecurity. At its core, cryptography has a mathematical foundation. Cryptographic encryption can be as simple as letter substitution or as complicated as modern methods used to secure networks and the Internet. Students encode and decode messages using encryption methods, and explore different levels of complexity used to hide or secure information. | | |
| | | | | | |
| For example, students could identify methods of secret communication used during the Revolutionary War (e.g., ciphers, secret codes, invisible ink, hidden letters) and then secure their own methods such as substitution ciphers or steganography (i.e., hiding messages inside a picture or other data) to compose a message from either the Continental Army or British Army. (HSS.8.1) | | | | | |
| | | | | | |
| Alternatively, students could explore functions and inverse functions for encryption and decryption and consider functions that are complex enough to keep data secure from their peers. (CA CCSS for Mathematics 8.F.1) | Networks & the Internet | Cybersecurity | Abstraction | 4.4 | |
| 6-8 | 6-8.DA.7 | Represent data in multiple ways. | Computers store data as sequences of 0s and 1s (bits). Software translates to and from this low-level representation to higher levels that are understandable by people. Furthermore, higher level data can be represented in multiple ways, such as the digital display of a color and its corresponding numeric RGB value, or a bar graph, a pie chart, and table representation of the same data in a spreadsheet. | | |
| | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| For example, students could use a color picker to explore the correspondence between the digital display or name of a color (high-level representations) and its RGB value or hex code (low-level representation). | | | | | |
| | | | | | |
| Alternatively, students could translate a word (high-level representation) into Morse code or its corresponding sequence of ASCII codes (low-level representation). | | | | | |
| | | | | | |
| | Data & Analysis | Storage | Abstraction | 4.4 | |
| 6-8 | 6-8.DA.8 | Collect data using computational tools and transform the data to make it more useful. | Data collection has become easier and more ubiquitous. The cleaning of data is an important transformation for ensuring consistent format, reducing noise and errors (e.g., removing irrelevant responses in a survey), and/or making it easier for computers to process. Students build on their ability to organize and present data visually to support a claim, understanding when and how to transform data so information can be more easily extracted. Students also transform data to highlight or expose relationships. | | |
| | | | | | |
| For example, students could use computational tools to collect data from their peers regarding the percentage of time technology is used for school work and entertainment, and then create digital displays of their data and findings. Students could then transform the data to highlight relationships representing males and females as percentages of a whole instead of as individual counts. (CA CCSS for Mathematics 6.SP.4, 7.SP.3, 8.SP.1, 8.SP.4) | | | | | |
| | | | | | |
| Alternatively, students could collect data from online forms and surveys, from a sensor, or by scraping a web page, and then transform the data to expose relationships. They could highlight the distribution of data (e.g., words on a web page, readings from a sensor) by giving quantitative measures of center and variability. (CA CCSS for Mathematics 6.SP.5.c, 7.SP.4) | Data & Analysis | Collection Visualization & Transformation | Communicating | 7.1 | |

| | | | | | |
|---|---|---|---|---|---|
| 6-8 | 6-8.DA.9 | Test and analyze the effects of changing variables while using computational models. | Variables within a computational model may be changed, in order to alter a computer simulation or to more accurately represent how various data is related. Students interact with a given model, make changes to identified model variables, and observe and reflect upon the results. | | |
| | | | | | |
| For example, students could test a program that makes a robot move on a track by making changes to variables (e.g., height and angle of track, size and mass of the robot) and discussing how these changes affect how far the robot travels. (CA NGSS: MS-PS2-2) | | | | | |
| | | | | | |
| Alternatively, students could test a game simulation and change variables (e.g., skill of simulated players, nature of opening moves) and analyze how these changes affect who wins the game. (CA NGSS: MS-ETS1-3) | | | | | |
| | | | | | |
| Additionally, students could modify a model for predicting the likely color of the next pick from a bag of colored candy and analyze the effects of changing variables representing the common color ratios in a typical bag of candy. (CA CCSS for Mathematics 7.SP.7, 8.SP.4) | Data & Analysis | Inference & Models | Abstraction, Testing | 4.4, 6.1 | |
| 6-8 | 6-8.AP.10 | Use flowcharts and/or pseudocode to design and illustrate algorithms that solve complex problems. | Complex problems are problems that would be difficult for students to solve without breaking them down into multiple steps. Flowcharts and pseudocode are used to design and illustrate the breakdown of steps in an algorithm. Students design and illustrate algorithms using pseudocode and/or flowcharts that organize and sequence the breakdown of steps for solving complex problems. | | |
| | | | | | |
| For example, students might use a flowchart to illustrate an algorithm that produces a recommendation for purchasing sneakers based on inputs such as size, colors, brand, comfort, and cost. | | | | | |
| | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| Alternatively, students could write pseudocode to express an algorithm for suggesting their outfit for the day, based on inputs such as the weather, color preferences, and day of the week. | Algorithms & Programming | Algorithms | Abstraction | 4.4, 4.1 | |
| 6-8 | 6-8.AP.11 | Create clearly named variables that store data, and perform operations on their contents. | A variable is a container for data, and the name used for accessing the variable is called the identifier. Students declare, initialize, and update variables for storing different types of program data (e.g., text, integers) using names and naming conventions (e.g. camel case) that clearly convey the purpose of the variable, facilitate debugging, and improve readability. | | |
| | | | | | |
| For example, students could program a quiz game with a score variable (e.g. quizScore) that is initially set to zero and increases by increments of one each time the user answers a quiz question correctly and decreases by increments of one each time a user answers a quiz question incorrectly, resulting in a score that is either a positive or negative integer. (CA CCSS for Mathematics 6.NS.5) | | | | | |
| | | | | | |
| Alternatively, students could write a program that prompts the user for their name, stores the user's response in a variable (e.g. userName), and uses this variable to greet the user by name. | Algorithms & Programming | Variables | Creating | 5.1, 5.2 | |
| 6-8 | 6-8.AP.12 | Design and iteratively develop programs that combine control structures and use compound conditions. | Control structures can be combined in many ways. Nested loops are loops placed within loops, and nested conditionals allow the result of one conditional to lead to another. Compound conditions combine two or more conditions in a logical relationship (e.g., using AND, OR, and NOT). Students appropriately use control structures to perform repetitive and selection tasks. | | |
| | | | | | |
| 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. (CA CCSS for ELA/Literacy W.6.3, W. 7.3, W.8.3) | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| Alternatively, students could use compound conditionals when writing a program to test whether two points lie along the line defined by a particular linear function. (CA CCSS for Mathematics 8.EE.7) | | | | | |
| | | | | | |
| Additionally, students could use nested loops to program a character to do the "chicken dance" by opening and closing the beak, flapping the wings, shaking the hips, and clapping four times each; this dance "chorus" is then repeated several times in its entirety. | Algorithms & Programming | Control | Creating | 5.1, 5.2 | |
| 6-8 | 6-8.AP.13 | Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. | Decomposition facilitates 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. Students break down (decompose) problems into subproblems, which can be further broken down to smaller parts. | | |
| | | | | | |
| Students could create an arcade game, with a title screen, a game screen, and a win/lose screen with an option to play the game again. To do this, students need to identify subproblems that accompany each screen (e.g., selecting an avatar goes in the title screen, events for controlling character action and scoring goes in the game screen, and displaying final and high score and asking whether to play again goes in the win/lose screen). | | | | | |
| | | | | | |
| Alternatively, students could decompose the problem of calculating and displaying class grades. Subproblems might include: accept input for students grades on various assignments, check for invalid grade entries, calculate per assignment averages, calculate per student averages, and display histograms of student scores for each assignment. (CA CCSS for Mathematics 6.RP.3c, 6.SP.4, 6.SP.5) | Algorithms & Programming | Modularity | Computational Problems | 3.2 | |

| | | | | | |
|---|---|---|---|---|---|
| 6-8 | 6-8.AP.14 | Create procedures with parameters to organize code and make it easier to reuse. | Procedures support modularity in developing programs. Parameters can provide greater flexibility, reusability, and efficient use of resources. Students create procedures and/or functions that are used multiple times within a program to repeat groups of instructions. They generalize the procedures and/or functions by defining parameters that generate different outputs for a wide range of inputs. | | |
| | | | | | |
| For example, students could create a procedure to draw a circle which involves many instructions, but all of them can be invoked with one instruction, such as “drawCircle.” By adding a radius parameter, students can easily draw circles of different sizes. (CA CCSS for Mathematics 7.G.4) | | | | | |
| | | | | | |
| Alternatively, calculating the area of a regular polygon requires multiple steps. Students could write a function that accepts the number and length of the sides as parameters and then calculates the area of the polygon. This function can then be re-used inside any program to calculate the area of a regular polygon. (CA CCSS for Mathematics 6.G.1) | Algorithms & Programming | Modularity | Abstraction | 4.1, 4.3 | |
| 6-8 | 6-8.AP.15 | Seek and incorporate feedback from team members and users to refine a solution that meets user needs. | Development teams that employ user-centered design processes create solutions (e.g., programs and devices) that can have a large societal impact (e.g., an app that allows people with speech difficulties to allow a smartphone to clarify their speech). Students 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, or color contrast. | | |
| | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| For example, if students are designing an app to teach their classmates about recycling, they could first interview or survey their classmates to learn what their classmates already know about recycling and why they do or do not recycle. After building a prototype of the app, the students could then test the app with a sample of their classmates to see if they learned anything from the app and if they had difficulty using the app (e.g., trouble reading or understanding text). After gathering interview data, students could refine the app to meet classmate needs. (CA NGSS: MS-ETS1-4) | Algorithms & Programming | Program Development | Collaborating, Inclusion | 2.3, 1.1 | |
| 6-8 | 6-8.AP.16 | Incorporate existing code, media, and libraries into original programs, and give attribution. | Building on the work of others enables students to produce more interesting and powerful creations. Students use portions of code, algorithms, digital media, and/or data created by others in their own programs and websites. They give attribution to the original creators to acknowledge their contributions. | | |
| | | | | | |
| 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. | | | | | |
| | | | | | |
| Alternatively, when creating a website to demonstrate their knowledge of historical figures from the Civil War, students may use a professionally-designed template and public domain images of historical figures. (HSS. 8.10.5) | | | | | |
| | | | | | |
| Additionally, students could import libraries and connect to web application program interfaces (APIs) to make their own programming processes more efficient and reduce the number of bugs (e.g., to check whether the user input is a valid date, to input the current temperature from another city). | Algorithms & Programming | Program Development | Abstraction, Creating, Communicating | 4.2, 5.2, 7.3 | |

| | | | | | |
|---|---|---|---|---|---|
| 6-8 | 6-8.AP.17 | Systematically test and refine programs using a range of test cases. | Use cases and test cases are created to evaluate whether programs function as intended. At this level, students develop use cases and test cases with teacher guidance. Testing should become a deliberate process that is more iterative, systematic, and proactive than at lower levels. | | |
| | | | | | |
| For example, students 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). | | | | | |
| | | | | | |
| Alternatively, in an interactive program, students could test that the character cannot move off of the screen in any direction, cannot move through walls, and can interact with other characters. They then adjust character behavior as needed. | Algorithms & Programming | Program Development | Testing | 6.1 | |
| 6-8 | 6-8.AP.18 | Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts. | 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 assume pre-defined roles within their teams and manage the project workflow using structured timelines. With teacher guidance, they begin to create collective goals, expectations, and equitable workloads. | | |
| | | | | | |
| For example, students could decompose 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. | | | | | |
| | | | | | |
| Alternatively, students could work as a team to develop a storyboard for an animation representing a written narrative, and then program the scenes individually. (CA CCSS for ELA/Literacy W.6.3, W.7.3, W.8.3) | Algorithms & Programming | Program Development | Collaborating, Creating | 2.2, 5.1 | |
| 6-8 | 6-8.AP.19 | Document programs in order to make them easier to use, read, test, and debug. | | | |

| | | | | | |
|---|---|---|---|---|---|
| | Documentation allows creators, end users, and other developers to more easily use and understand a program. Students provide documentation for end users that explains their artifacts and how they function (e.g., project overview, user instructions). They also include comments within code to describe portions of their programs and make it easier for themselves and other developers to use, read, test, and debug. | | | | |
| | | | | | |
| For example, students could add comments to describe functionality of different segments of code (e.g., input scores between 0 and 100, check for invalid input, calculate and display the average of the scores). They could also communicate the process used by writing design documents, creating flowcharts, or making presentations. (CA CCSS for ELA/Literacy SL.6.5, SL.7.5, SL.8.5) | Algorithms & Programming | Program Development | Communicating | 7.2 | |
| 6-8 | 6-8.IC.20 | Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options. | Advancements in computer technology are neither wholly positive nor negative. However, the ways that people use computing technologies have tradeoffs. Students consider current events related to broad ideas, including privacy, communication, and automation. | | |
| | | | | | |
| For example, students could compare and contrast the impacts of computing technologies with the impacts of other systems developed throughout history such as the Pony Express and US Postal System. (HSS.7.8.4) | | | | | |
| | | | | | |
| Alternatively, students could identify tradeoffs for both personal and professional uses of a variety of computing technologies. For instance, driverless cars can increase convenience and reduce accidents, but they may be susceptible to hacking. The emerging industry will reduce the number of taxi and shared-ride drivers, but may create more software engineering and cybersecurity jobs. | Impacts of Computing | Culture | Communicating | 7.2 | |

| | | | | | |
|---|---|---|---|---|---|
| 6-8 | 6-8.IC.21 | Discuss issues of bias and accessibility in the design of existing technologies. | Computing technologies should support users of many backgrounds and abilities. In order to maximize accessiblity, these differences need to be addressed by examining diverse populations. With the teacher's guidance, students test and discuss the usability of various technology tools, such as apps, games, and devices. | | |
| | | | | | |
| For example, students could discuss the impacts of facial recognition software that works better for lighter skin tones and recognize that the software was likely developed with a homogeneous testing group. Students could then discuss how accessibility could be improved by sampling a more diverse population. (CA CCSS for ELA/Literacy SL.6.1, SL.7.1, SL.8.1) | Impacts of Computing | Culture | Inclusion | 1.2 | |
| 6-8 | 6-8.IC.22 | Collaborate with many contributors when creating a computational artifact. | Users have diverse sets of experiences, needs, and wants. These need to be understood and integrated into the design of computational artifacts. Students use applications that enable crowdsourcing to gather services, ideas, or content from a large group of people. At this level, crowdsourcing can be done at the local level (e.g., classroom, school, or neighborhood) and/or global level (e.g., age-appropriate online communities). | | |
| | | | | | |
| For example, a group of students could use electronic surveys to solicit input from their neighborhood regarding an important social or political issue. They could collaborate with a community artist to combine animations and create a digital community collage informing the public about various points of view regarding the topic. (VAPA Visual Art 8.5.2, 8.5.4) | Impacts of Computing | Social Interactions | Collaborating, Creating | 2.4, 5.2 | |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| 6-8 | 6-8.IC.23 | Compare tradeoffs associated with licenses for computational artifacts to balance the protection of the creators' rights and the ability for others to use and modify the artifacts. | Using and building on the works of others allows people to create meaningful works and fosters innovation. Copyright is an important law that helps protect the rights of creators so they receive credit and get paid for their work. Creative Commons is a kind of copyright that makes it easier for people to copy, share, and build on creative work, as long as they give credit for it. There are different kinds of Creative Commons licenses that allow people to do things such as change, remix, or make money from their work. As creators, students can pick and choose how they want their work to be used, and then create a Creative Commons license that they include in their work. |  |  |
|  |  |  |  |  |  |
| For example, students could create interactive animations to educate others on bullying or protecting the environment. They then select an appropriate license to reflect how they want their program to be used by others (e.g., allow others to use their work and alter it, as long as they do not make a profit from it). Students use established methods to both protect their artifacts and attribute use of protected artifacts. | Impacts of Computing | Safety Law & Ethics | Communicating | 7.3 |  |
| 6-8 | 6-8.IC.24 | Compare tradeoffs between allowing information to be public and keeping information private and secure. | While it is valuable to establish, maintain, and strengthen connections between people online, security attacks often start with intentionally or unintentionally providing personal information online. Students identify situations where the value of keeping information public outweighs privacy concerns, and vice versa. They also recognize practices such as phishing and social engineering and explain best practices to defend against them. |  |  |
|  |  |  |  |  |  |
| For example, students could discuss the benefits of artists and designers displaying their work online to reach a broader audience. Students could also compare the tradeoffs of making a shared file accessible to anyone versus restricting it to specific accounts. (CA CCSS for ELA/Literacy SL.6.1, SL.7.1, SL.8.1) |  |  |  |  |  |
|  |  |  |  |  |  |

| | | | | | |
|---|---|---|---|---|---|
| Alternatively, students could discuss the benefits and dangers of the increased accessibility of information available on the internet, and then compare this to the advantages and disadvantages of the introduction of the printing press in society. (HSS.7.8.4) | Impacts of Computing | Safety Law & Ethics | Communicating | 7.2 | |