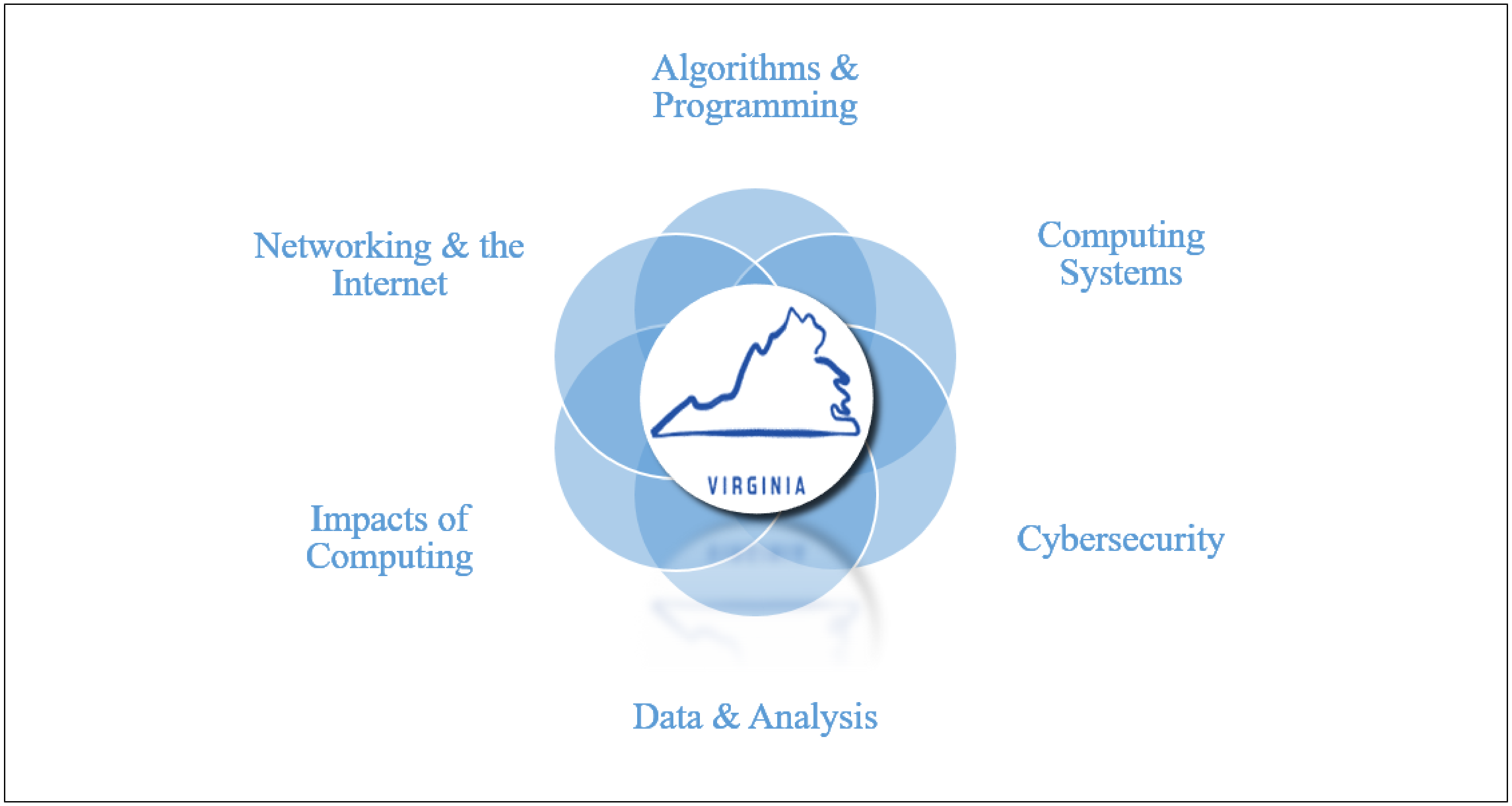
Computer Science Standards of Learning
Curriculum Framework



Board of Education
Commonwealth of Virginia

The 2017 *Computer Science Curriculum Framework* can be found on the Virginia Department of Education's Web site.

Introduction

The *Computer Science Standards of Learning* Curriculum Framework amplifies the *Computer Science Standards of Learning for Virginia Public Schools* and defines the content knowledge, skills, and understandings that are measured by the Standards of Learning. The Computer Science Curriculum Framework provides additional guidance to school divisions and their teachers as they develop an instructional program appropriate for their students. It assists teachers as they plan their lessons by identifying essential questions and vocabulary to drive instruction and defining the essential skills students should demonstrate. This supplemental framework delineates in greater specificity the minimum content that all teachers should teach and all students should learn.

School divisions should use the *Computer Science Curriculum Framework* as a resource for developing sound curricular and instructional programs. This framework should not limit the scope of instructional programs. Additional knowledge and skills that can enrich instruction and enhance students' understanding of the content identified in the Standards of Learning should be included as part of quality learning experiences.

Each topic in the *Computer Science Standards of Learning* Curriculum Framework is developed around the Standards of Learning. The format of the Curriculum Framework facilitates teacher planning by broadening the context of the standards and identifying essential student skills that should be the focus of instruction for each standard.

Context of the Standard
The Context of the Standard provides educators an explanation of the standard, including a description and the vertical development of the concept. This context will support teachers in incorporating computer science content into discipline-specific lessons. The intention of the Computer Science standards in grades K-8 is that Computer Science principles be integrated throughout content area instruction.

Essential Skills
The Essential Skills define student performance expectations aligned to each standard. The intent of the K-8 computer science standards is that the concepts are integrated into existing disciplines and this will result in these skills being emphasized differently in each content area. The expectation is that these Essential Skills are partnered with content area performance expectations as appropriate in instruction. At the high school level, the expectations in the 2017 *Computer Science Standards of Learning Curriculum Framework* are to be used in the support of standalone computer courses; the essential skills outlined in the document are not intended to be integrated into other coursework unless a teacher chooses to use the content to support discipline practices.

Essential Questions
Each standard has identified key questions to drive classroom instruction. These questions lead teachers and students toward the big ideas of each concept and provide a more holistic viewpoint used to lead instruction relating to the context of each standard.

Essential Vocabulary
In order to effectively communicate Computer Science concepts, essential vocabulary terms are defined in grade-level appropriate terms. These definitions are found in the glossary (Appendix A).

Computer Science Foundations

The Computer Science Foundations standards outline the content for a one-year course with an emphasis on computer programming within the context of broader concepts of computer science. The standards build on the concepts of computer science developed in prior grade levels. The standards provide a transition from block-based programming to a text-based programming language and familiarize the student with developing and executing computer programs. Teachers are encouraged to select programming languages and environments, problems, challenges, and activities that are appropriate for their students to successfully meet the objectives of the standards.

Programmable computing tools will be used to facilitate design, analysis, and implementation of computer programs. Students should use these tools for exploring and creating computer programs, facilitating reasoning and problem solving, and verifying solutions.

Computing Systems

CSF.1 The student will

a. compare the structures, functions, and interactions between application software, system software, and hardware; and
b. explore the relationship between hardware and software using the Internet of Things.

Context of the Standard
Computing systems are comprised of many modular, interconnected pieces of hardware and software. These pieces work together in a hierarchical system to accomplish complex tasks. When computers communicate with one another over networks, data is passed from application software, down to system software, and finally down to a hardware layer that communicates directly with the hardware layer of the receiving machine. The receiving machine then passes data up to its own system software and application software repeating the process in reverse. Often, any piece of data will pass up and down the hardware and software layers of many machines before reaching its final destination (see CSF.2). Internet-connected devices (such as Internet of Things devices) are a good example of this phenomenon because the device itself will often relay data to another device that controls its behavior. For example, an IoT thermostat (hardware) will push temperature data to an application (software) running on a smartphone. In turn, the application on the smartphone will control the thermostat at the user's request by sending data over the Internet.

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills: • Illustrate the hierarchy of hardware, system software, and application software. • Describe the relationship between hardware and software. • Analyze the path that data travels along in a typical instance of networked communication.	Students should *investigate* these concepts: • What are the roles of hardware and software when using a network? • Why do computing systems make use of a layered hierarchy when communicating with other systems?	Students should *apply* these terms in context: • Hardware • Software (System & Application) • Network • Internet of Things (IoT) • Hierarchy

Networks and the Internet

CSF.2 The student will 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.

Context of the Standard
Computers can send and receive messages when connected to one another. These networked devices pass information among themselves using a specific set of rules called a protocol. Reliable communication through networks depends on all sent information arriving at its destination, regardless of heavy traffic or damaged connections. Computers also need a reliable way of reporting and resolving communication errors. To achieve these two prerequisites, computers break down messages (e.g., emails, images, music) into smaller chunks of data called packets. These packets make their way to their destination computer separately using different routes, after which the receiving computer puts them back together in order. In large networks like the Internet, packets may travel among dozens of different computing devices; each device passes them on until all of the packets reach their destination. If a packet is missing and the message is incomplete, the receiving computer will request that the sender resubmit the message.

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills: • Describe the process of sending a file through a network. • Create a model of networked communication. • Explain how sending data in packets ensures reliable communication among computing devices.	Students should *investigate* these concepts: • What are the physical qualities of a reliable network? • What are some ways networked communication might be disrupted?	Students should *apply* these terms in context: • Internet • Network • Packet • Protocol • Reliability

CSF.3 The student will explain the role of protocols in transmitting data across networks and the Internet.

Context of the Standard
Even the sending of a simple email message requires multiple levels of communication between devices. In order for a message to maintain its fidelity, it is essential that there be a series of protocols, or agreed upon standards of syntax, format, packet size, etc, used at each level. Anyone can join at any time and successfully communicate with anyone else on the network without deciding on rules for communication in advance. Internet communication is made possible by several protocols, including TCP, IP, UDP, and DNS. These sets of rules interact to build a model used by almost all Internet communication (OSI model). Without them, it would be very difficult for computing devices to communicate with one another without explaining in advance how to interpret information sent over the network.

Essential Skills	Essential Questions	Essential Vocabulary
Students should *demonstrate* these skills: • Define *protocol* in the context of computer science. • Summarize how protocols make a decentralized Internet possible. • Explain the role of protocols in networked communication.	Students should *investigate* these concepts: • What are some examples of protocols that humans use to communicate with one another? • How might the Internet function differently if every nation had its own set of protocols?	Students should *apply* these terms in context: • Network • Packet • Internet • Reliability • Scalability • Protocol

CSF.4 The student will evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing.

Context of the Standard
Networks are made up of interconnected computing devices. Some are familiar, like laptops, smartphones, and desktop computers. Others, called servers, are designed to send files over the Internet. Servers are Internet-connected computing devices that store web pages and other files. They process requests from computers and then distribute results over the Internet. The computing devices in a network work together to ensure that communication is fast and reliable. How they are arranged in that network is referred to as the network topology. Many people have computing networks in their homes, where several devices are connected to the same access point. This small home network is called a local area network (LAN). Each device in a LAN has a unique address called an IP (Internet protocol) address. Neighborhoods might have dozens of local area networks, one or more for each home. Just like the devices within it, each local area network has a unique IP address. Local area networks are connected to switches, which serve as hubs for LANs in a neighborhood. This web of LANs is called a wide area network (WAN). A city might have hundreds of wide area networks, each with its own unique IP address. Wide area networks all eventually connect to routers. The Internet is a vast web of routers, each one passing along packets (see CSF.2 & CSF.3) to the appropriate destination.

Context of the Standard
This system is reliable and scalable because each machine only has to keep track of the devices it is directly connected to. A smartphone doesn't need to know the IP addresses of all the switches and routers between it and the server it wants to connect to; it only has to know the IP address of the server (stored in an easily accessible public database) and the IP address of its local wireless access point.

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills: • Model how information is transmitted among computing devices that make a up a network. • Evaluate how addressing facilitates the scalability and reliability of the Internet. • Illustrate the arrangement or topology among elements of a network.	Students should *investigate* these concepts: • How many devices might a packet travel through on the way to its destination? • Are Internet communications public or private? How do you know?	Students should *apply* these terms in context: • IP address • Local Area Network (LAN) • Reliability • Router • Scalability • Server • Switch • Wide Area Network (WAN)

Cybersecurity

CSF.5 The student will identify and explain ways that sensitive data (assets) can be threatened by malware and other computer attacks, using appropriate terminology.

Context of the Standard
People can exploit vulnerabilities in computing devices, systems, and use cases with malicious intent. Some exploits will focus on disrupting network services, while others are perpetrated with the goal of gaining access to sensitive assets. These people make use of malware (malicious software installed on a computer) and viruses (scripts that may be hidden in existing files or programs) to spy on users, access private files and data, or lock victims out of their computers. For example, ransomware is malicious software that encrypts a victim's files and demands a ransom before the data is decrypted. In contrast, a virus might create a backdoor in a system that allows an attacker to gain access to it at a later date. Attackers might also monitor network activity, recording email addresses, phone numbers, and other unprotected data to use in future attacks. For example, someone could set up a fake wireless access point, and will exploit the privileges afforded to a trusted network access point to spy on data transmitted over that network connection. Attackers will also make more mundane attacks; they may guess passwords to gain access to emails, make fake phone calls to gain information about victims, or send emails with fake links or viruses stored in seemingly safe attachments. The strategies people use to exploit computing systems are always evolving, and security experts are tasked with understanding how to adapt to new attacks and vulnerabilities.

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills: • Identify different types of cyber-attacks. • Explain why attackers might want to gain personal information/sensitive data from their target. • Describe different cyber-attacks and describe how they might affect a computing system.	Students should *investigate* these concepts: • How might a security expert protect data on a network? • If someone discovers a vulnerability in a piece of software, should they make it public, or keep it secret?	Students should *apply* these terms in context: • Data • Exploit • Malware • Virus • Vulnerability

CSF.6 The student will give examples of ways to protect sensitive data (assets) from malware and other computer attacks and evaluate them according to multiple criteria.

Context of the Standard
Bad actors, or people who use computers with malicious intent, make use of many strategies to gain access to personal data. People seeking to protect their data can take several proactive steps which ensure that their data is safe from common attacks. Creating secure passwords unique to every account, configuring the firewalls on personal computing devices, not connecting to unfamiliar wireless access points, installing reputable anti-virus and anti-malware software (anti-x software), and practicing safe-decision making online by not clicking unfamiliar links or responding to unsolicited communication are all good strategies for protecting oneself online. However, there is no way to ensure that any online communication is private with 100% confidence. Part of protecting oneself online is knowing that all online communication is potentially public, and behaving accordingly. In evaluating possible protective measures, the possible threats should be considered as well as system vulnerabilities and risks of breach. Most people will not be victims of large-scale data breaches. Highly visible or vulnerable individuals attract more attention from bad actors; politicians, corporate executives, and celebrities must often take extraordinary steps to adequately protect their data from potential exploitation. People who fail to protect themselves from attacks will often be attacked multiple times, as their personal information is disseminated online.

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills: • Compare and contrast safe and unsafe computing practices. • Describe ways to protect personal information on a computing device. • Synthesize and communicate strategies for protecting personal data	Students should *investigate* these concepts: • What are some strategies for protecting your sensitive data? • Who are some people who might be popular targets for cyber-attack or espionage?	Students should *apply* these terms in context: • Anti-X Software • Cybersecurity • Cyber Attack • Data • Vulnerability

Essential Skills	Essential Questions	Essential Vocabulary
to someone unfamiliar with cyber best practices. • Evaluate a protective measure in its effectiveness against cyberthreats.		• Firewall

CSF.7 The student will explain typical tradeoffs between usability and security and recommend security measures in a given scenario based on these (or other) tradeoffs.

Context of the Standard
The people who use computing devices are often the most significant vulnerability in a system. Computing devices have their own host of security vulnerabilities, but users can be tricked, misled, or otherwise manipulated into granting bad actors access to their personal data. When interacting with computing systems, users experience a tradeoff between security and usability. If a computing devices makes use of multiple layers of security, users may begin to encounter barriers that make the computer more difficult to use. For example, a highly filtered Internet connection will keep users from accessing malicious websites, but may also inadvertently block some safe websites. False positives like these are inevitable in a highly secure system. Anti-virus software will often scan downloaded files for malicious content, forcing users to wait until the scan is done before they open the file. High levels of security can create significant barriers to usability; as a result, security measures are often calibrated to match the data being secured. For example, an online banking profile is more secure (and as a result, will be less usable) than an online gaming account. To ensure that sensitive data is not accessed by unwanted people, users often must provide private credentials (i.e., passwords) when accessing their information. When tasked with protecting particularly sensitive data, software designers will often add additional layers of security, like security questions or two-factor authentication (i.e., when a user is required to produce two sets of credentials, like a password and a secret code texted to their phone).

Essential Skills	Essential Questions	Essential Vocabulary
Students should *demonstrate* these skills: • Explain how security measures might affect information transmission speed within a network. • Investigate and describe how security measures might be adjusted for different types of information. • Identify different strategies for securing sensitive information.	Students should *investigate* these concepts: • What are some examples of software you use that require credentials? What information are these credentials protecting? • Do you experience any tradeoffs between usability and security?	Students should *apply* these terms in context: • Authentication • Credentials • Malicious • Tradeoff • Useability

CSF.8 The student will write or adapt a program to validate its input and to avoid certain kinds of vulnerabilities.

Context of the Standard
People who want to compromise a system will often exploit vulnerabilities built into software by misusing or exploiting existing avenues for user input. Examples of these types of attacks include SQL injection and command line injection, where attackers use existing software protocols to trick the program into revealing information about an underlying database or forcing the computer to execute code unintentionally. To protect against these kinds of attacks, designers will validate user input; in other words, designers will program the software to reject inappropriate input or fix it in predictable ways. Designers will often create a list of approved input values, or a list of input values that should be rejected. This ensures that the software is used only for its intended purpose. Another example of input validation is when software rejects insecure passwords during the account creation process. The software will tell users to pick a new password because the one they initially chose would be easy to guess or exploit.

Essential Skills	Essential Questions	Essential Vocabulary
Students should *demonstrate* these skills: • Design a program that takes user input and validates it against a list of approved input options. • Write a program that returns affirmative output given an input that is validated against an approved list.	Students should *investigate* these concepts: • What are some personal best practices for validating digital communication (e.g., emails or texts from unknown senders)? • Can you identify an example of input validation in the real world?	Students should *apply* these terms in context: • Command Line Injection • Exploit • Input • Software • SQL Injection • Validate • Vulnerability

Data and Analysis

CSF.9 The student will evaluate the tradeoffs in how data elements are organized and where data is stored.

Context of the Standard
On an individual program level, data are typically stored in some sort of a structure. These structures may include arrays, lists, or others. Arrays store multiple values in sequential memory and are referenced by an index or position. Multiple options are considered for the storage and access of these data. The choice of a data structure can be influenced by data access speed, memory usage, organizational complexity, and other factors. On an individual computer level, data are stored on local hard drives in a variety of file formats. Simple file formats typically use less memory space but only store basic forms of the data with little formatting. Larger file formats provide a variety of formatting options and extra protection but typically are only accessible through specific programs. On a network level, data are often stored on “cloud” servers and accessed remotely for manipulation and analysis. This is beneficial for work that needs to be conducted by multiple people, but requires network access to be possible. With consistent network access, there is need for data security and encryption to protect the data from unwanted access.

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills: • Identify different data structures used by programmers. • Compare and contrast the aspects of different data structures. • Compare and contrast local vs. cloud data storage. • Identify how data element storage differs depending on the program or system being used.	Students should *investigate* these concepts: • How can a programmer store multiple values in one structure? • What factors lead to choosing one structure over another one? • What are the benefits to using cloud storage instead of local storage? • What are the drawbacks to using cloud storage instead of local storage?	Students should *apply* these terms in context: • Array • List • Local storage • Cloud storage

CSF.10 The student will create interactive data visualizations using software tools to help others better understand real-world phenomena.

Context of the Standard
Software programs and hardware devices are constantly creating data. Smartwatches for example, generate data about movement, location, sleep, heart rate, and many other kinds of user input. In order to interpret the raw information generated by software and hardware devices, programmers create data visualizations—visual representations of data. Some visualizations are traditional (e.g., bar graphs, pie charts, scatter plots), while others are more dynamic, including sounds, haptic feedback, or input which allow users to explore the data interactively. Some programmers will use data to create data-based art, which focuses less on communicating trends and more on creating evocative visual, sounds, or experiences. A visualization may even make use of data being gathered in real time, where sensors or programs update a database and the visualization program updates by pulling new data constantly. For example, a visualization might show domestic flights by pulling data from an air traffic control database.

Essential Skills	Essential Questions	Essential Vocabulary
Students should *demonstrate* these skills: • Generate or identify criteria for effective and/or evocative data visualization. • Create a visualization given a data set. • Use a data visualization to draw conclusions about the underlying raw data.	Students should *investigate* these concepts: • What types of data visualizations can you find on the Internet that is refreshed in real time (e.g., weather, traffic)? • What type of daily data could a school create, and how might you visualize that data?	Students should *apply* these terms in context: • Data Set • Data Visualization • Interactive Data • Raw Data

CSF.11 The student will use data analysis tools and techniques to identify patterns in data representing complex systems.

Context of the Standard
Software programs may generate incredibly large amounts of data. Software companies will use this data to make decisions about marketing, design, and future software updates. Scientists also use software to generate large amounts of data to better understand natural phenomena. The problem is that these large amounts of data are impossible to fully understand in a raw format, and too large and complex for simple graphical visualizations to communicate their depth. Data scientists and programmers will often use the term "big data" to refer to these massive datasets, which often contain many different variables and thousands or even millions of data points. In order to draw conclusions from large data sets and communicate those conclusions to stakeholders, programmers will create data visualizations (see CSF.10) that expand on or reimagine traditional data visualization methods (e.g., scatter plots, bar graphs, pie charts, line graphs). These data visualizations will often compare multiple parameters from a multivariate data set, or show how data changes over time using time-series data.

Essential Skills	Essential Questions	Essential Vocabulary
Students should *demonstrate* these skills: • Describe why it might be difficult to draw conclusions from big data. • Draw conclusions about large multivariate data sets given visualizations.	Students should *investigate* these concepts: • Is facial recognition an ethical use of data analysis? Why or why not? • What might you learn about a website from data that tracks where users look on the page?	Students should *apply* these terms in context: • Analysis • Big Data • Bivariate • Data Visualization • Multivariate • Time-Series

Algorithms and Programming
CSF.12 The student will develop a program working individually and in teams using a text-based language.

Context of the Standard
Programming is how people create new tools and experiences with computers. Programming languages are the creative medium of computer programming, allowing people to solve problems by generating new tools that run on computing devices. Many times, programmers work in teams by dividing the work among the team members. For example, each team member might work on one page of the application. This allows individuals to focus on making one aspect of the program as good as it can be, without worrying about the details of the rest of the application. After everyone has finished their pieces, the team will come together and collaboratively combine their pieces into a cohesive whole.

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills: • Organize a plan to develop a program individually and as part of a team. • Construct a tool that solves a problem using text-based programming. • Identify future possibilities for adding features or fixing issues in a given program.	Students should *investigate* these concepts: • Why might you test a product while building it rather than waiting until you finish the product? • What are some of the differences between programming alone versus programming with a team?	Students should *apply* these terms in context: • Iterative process • Program • Prototype • Text-based language

CSF.13 The student will identify the expected output of a program given a problem and some input.

Context of the Standard
Software programs take input from users, and then give the users feedback based on that input. Fundamentally, programming structures 1) take parameters, and 2) return values based on those parameters. If you created a program to add two numbers together, the user would specify the two numbers in question, and then the program would return the sum of the two values. Programmers elaborate on this logical structure to create software programs that give users complex, often multisensory feedback. The ability to analyze code and predict its output given an input value is a fundamental programming skill. Programmers will exercise this skill at all levels of expertise. Additionally, comparing the expected output of a program to the actual output of that program helps programmers educate themselves and learn about how they can edit their code and create software that works as expected. Programmers (especially beginners) should always be asking themselves: “What do I expect to happen when I execute this code, and what actually happened?” If a program returns an unexpected value, it often means that the programmer must go back and debug (repair) the program by re-evaluating their initial assumptions about how individual pieces of code function.

Essential Skills	Essential Questions	Essential Vocabulary
Students should *demonstrate* these skills: • Predict the output of a program that incorporates conditional statements given input values. • Create a program that returns given output values by taking given input values as parameters. • Repair a program that returns unexpected output values.	Students should *investigate* these concepts: • What questions should be considered when testing a program you wrote? • What are the next steps for you as a designer if the program does not work as expected?	Students should *apply* these terms in context: • Input • Output • Parameter • Return • Conditional • Execute • Debug

CSF.14 The student will design and iteratively develop programs for practical intent or personal expression, incorporating feedback from users.

Context of the Standard
Creating new software involves several steps, including designing, prototyping, testing, and debugging. Some programmers use the notion of a minimum viable product, or MVP, as a guide to help focus their creative efforts. An MVP is the simplest version of a tool that solves a problem. For example: if someone wants a tool to quickly move from place to place over long distances, a car might be a final solution. A car, however, is a very complex machine and may be difficult to design, prototype, and test in a reasonable amount of time. A minimum viable product for this problem might instead be a skateboard. It solves the problem, but is much easier to design, prototype, and test. After the skateboard is successful, the team might add on to it to solve the next problem (e.g., steering needs to be improved, so add a handle and make a scooter). This iterative process of designing, prototyping, testing, and debugging ensures that programmers and designers avoid wasting time and resources on a complex solution to a simple problem. For students learning programming, a minimum viable product is an excellent way of ensuring that students are frequently testing, debugging, and refining their software as they work toward short-term, iterative goals. By incorporating user feedback and testing throughout the build process from the very beginning, programmers ensure that they are not wasting time either working on code

Context of the Standard
that does not address users' problems, or writing code that will not work in the end. Programmers always try to hypothesize about an expected output, and then validate that hypothesis by testing code (CSF.13).

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills: • Design a program using a planning process (wireframe, logic flowchart, pseudocode). • Create a program based on a design plan starting with a MVP (Minimum Viable Product). • Demonstrate the iterative development process by participating and getting feedback from user testing and debugging.	Students should *investigate* these concepts: • How is feedback used through the iterative design process? • What is an example of an iterative process that we use in our regular, non-programming lives? • Who should programmers talk to before designing and programming an app? Why?	Students should *apply* these terms in context: • Debugging • Feedback/User Testing • Iterative Design • Minimum Viable Product • Prototype

CSF.15 The student will design and implement algorithms using

a. sequencing of instructions;
b. conditional execution; and
c. iteration.

Context of the Standard
An algorithm is a series of instructions for the computer to execute given some sort of input value. A simple algorithm might take in a series of numbers and return the sum of those numbers. More often, however, algorithms are more complicated. For example, a social media site might use inputs like browsing history, posts, comments, or video viewing history to decide which advertisements to show to a specific user. Software designers often create visual artifacts like flow charts that communicate the algorithmic process before they start writing code, in order to demonstrate the logical process the algorithm utilizes. Conditional execution refers to a part of an algorithm where the code asks the computer to compare two values, and execute two different pieces of code based on the relationship between those two values. For example, a sorting algorithm in a manufacturing setting might compare the weight of an object to a standard weight, and then sort that object based on whether it is heavier or lighter than the standard weight. Iteration means performing the same algorithm on a series of inputs in a loop. An algorithm might iterate over a long list of values, giving output for each of the values using the same algorithmic process.

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills: • Create a program that moves through a sequence of instructions when implemented. • Design a program that selects one of several values based on conditions using a flowchart.	Students should *investigate* these concepts: • What is the difference between instructions and an algorithm? • Think of an algorithm that you use in daily life. What are the input and output values? • What are the pros and cons for using an algorithm to solve a problem?	Students should *apply* these terms in context: • Algorithm • Conditional Execution • Flowchart • Iteration • List • Sequence

Essential Skills	Essential Questions	Essential Vocabulary
• Create an algorithm that iterates through a list of values producing output.		

CSF.16 The student will implement a program that accepts input values, stores them in appropriately named variables, and produces output.

Context of the Standard
An algorithm that accepts input values, stores them in variables, and produces output builds into a program (CSF.15). Input values can take many forms; clicks, mouse movement, physical orientation, typed words, voice, or images can all be quantified or otherwise digitally represented as input for an algorithm. Variables may be words, abbreviations, or symbols that store data. They allow coders to refer to data in a human-readable way, rather than as long strings of binary or hexadecimal numbers. A variable can be called almost anything, so coders will often use words that represent the data being stored in the variable. For example, a variable that stores weight data from a sensor might be called “weight” or “wgt.” Often, coders will choose names that allow code to be more or less read like a sentence, making it easier for colleagues and collaborators to understand what a given piece of code is supposed to do. Output can take many forms as well. Sometimes algorithms will output data designed to be processed by another computer program, while other times the output is intended to communicate data to human users. In the latter case, output might be visual (i.e., on a screen or light), haptic (e.g., vibrations), or aural (e.g., ringtones). Output is highly important; it makes the invisible work done by the computer visible to human users. Output is also used to help with debugging by giving programmers insights into how to fix programs that do not produce expected outputs.

Essential Skills	Essential Questions	Essential Vocabulary
Students should *demonstrate* these skills: • Develop a program that takes user input which is then stored in a variable. • Create appropriately named variables to store input data from a user. • Produce output made from various stored variables using both strings and integers (e.g. concatenation).	Students should *investigate* these concepts: • What factors should you consider when naming variables? • Are there types of information/data that can't be stored in a variable? Explain. • Think of an app you use. What type(s) of output does it generate?	Students should *apply* these terms in context: • Concatenation • Integer • Input • Output • String • Variable

CSF.17 The student will trace the execution of an algorithm, illustrating output and changes in values of named variables.

Context of the Standard
Code tracing is the practice of analyzing a section of code and identifying the values of critical variables at certain points in the algorithmic process. Tracing is the first step toward debugging or otherwise decoding a computational process; it involves making hypotheses about how the code will change input values during execution, and ultimately involves predicting the output of an algorithmic process (CSF.13). As programmers trace code using a trace table, they will identify how different variables change over time, and then compare that hypothesis to outputs given by the program. If the expected output differs from the actual output, programmers will either update their hypothesis or change the code to reflect their intended design.

Essential Skills	Essential Questions	Essential Vocabulary
Students should *demonstrate* these skills:	Students should *investigate* these concepts:	Students should *apply* these terms in context:

Essential Skills	Essential Questions	Essential Vocabulary
• Construct a trace table that follows the changes of variables in an algorithm. • Explain the output of a given algorithm, given different input values.	• What situations call for a programmer to use code tracing? • What are other techniques might programmers use to trace the change in variables while creating code?	• Algorithm • Code Tracing • Execution of code • Trace Table

CSF.18 The student will apply the basic operations used with numeric and non-numeric data types in developing programs.

Context of the Standard
Operators are mathematical symbols which ask computers to perform specific tasks given some sort of input. Most operators are fairly straightforward, and they generally fall into one of three categories: arithmetic, Boolean, or logical. Arithmetic operators mostly involve processing numerical data. They take in multiple inputs (i.e., numbers) and generate one output depending on the operator being used. Arithmetic operators include addition (+), subtraction (-), multiplication (*), and division (/). For example, a computer would produce “4” given the input “2 + 2.” The operator (+) tells the computer to sum the two input numbers. Unlike arithmetic operators, which produce numerical outputs, Boolean operators produce an output of either “true” or “false.” Boolean operators include “==”, “<”, “>”, “<=”, “>=”, “!=”, and more. For example, the expression “2 < 4” would return a value of “true,” because 2 is indeed less than 4. The expression “5 == 6” would return a value of “false,” because 5 does not equal six. Boolean operators may be notated differently in different programming languages, so it is wise to consult the language documentation if Boolean operators do not return expected values. Logical operators are used to combine Boolean expressions to allow for more complex processing. Logical operators include “and” (&&) and “or” (||), among others. Logical operators return Boolean values (true or false). The expression “5 == 6 && 4 == 4” would return a value of false, because only one of the Boolean expressions returned a value of true. In contrast, the expression “5 == 6 || ‘word’ == ‘word’ ” would return a value of “true” because the “or” operator (||) returns a value of “true” if any one of the

Context of the Standard
expressions returns “true.” Like Boolean operators, logical operators differ among programming languages (though there is some overlap).

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills: • Create an algorithm that changes the value of a named variable using arithmetic operations. • Design a code segment that uses a Boolean operator and a logical operation as part of a conditional statement.	Students should *investigate* these concepts: • How is Boolean logic used to accomplish a task? • What types of questions should not be answered with a Boolean response?	Students should *apply* these terms in context: • Arithmetic Operators • Boolean Operators • Conditional Statement • Logical Operators

CSF.19 The student will use predefined functions to simplify the solution of a complex problem.

Context of the Standard
Functions are like variables, except instead of storing data they store lines of code. Programmers use functions to help “chunk” large, self-contained sections of code to help make code more readable and to allow code to be used multiple times without having to re-write it. Most functions take input, and give some sort of output value. Functions almost always have parentheses after them; for example, a function that sums a list of values might look like this: “sum().” To use this function, you would put the values you would like to sum in the parentheses: “sum(2, 6, 7, 2, 1).” Given this input, the function would produce “18” as an output. Many programming languages come with predefined functions that programmers can use without having to worry about the underlying code that makes them work. For example, Math.random() returns a random number between 0 and 1 (in Java). Some

Context of the Standard
programming languages have more predefined functions than others. In addition to the language-specific predefined functions, programmers may download packages that contain even more predefined functions, extending the library of functions made available by the language code base. For example, SciKitLearn is a Python library package that provides predefined functions for data analysis and machine learning.

Essential Skills	Essential Questions	Essential Vocabulary
Students should *demonstrate* these skills: • Solve a problem by calling premade functions in a code segment or program. • Explain the outputs of common premade functions in a given programming language.	Students should *investigate* these concepts: • Think of a multi-step process in your daily life that has a single name (e.g. make dinner). What are the input and output values of this process? • What roles do functions serve in computer programs?	Students should *apply* these terms in context: • Function • Library • API

CSF.20 The student will apply simple algorithms to a collection of data.

Context of the Standard
Algorithms are a powerful tool for analyzing data. There are many ways for algorithms to process data; most of these examples are specific to a given domain or problem. For example, a programmer might create an algorithm that processes large amounts of climate data, producing an output that summarizes the impact CO2 emissions have had on global temperatures over time. Sorting algorithms are a simpler example; they take in a large list of data, and return that same data sorted into categories. Sorting algorithms are often part of much larger computational systems, but are also an example of data processing that is accessible to novice programmers building their first algorithm.

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills: • Describe the purpose of a set of code containing an algorithm. • Create a program that applies a simple algorithm to items in a list.	Students should *investigate* these concepts: • What are some reasons to use algorithms for analyzing data, rather than using other means? • Who is responsible for the actions of a biased algorithm?	Students should *apply* these terms in context: • Algorithm • Array • Data • List

CSF.21 The student will create programs

a. demonstrating an understanding that program development is an ongoing process that requires adjusting and debugging along the way; and
b. using version control to create and refine programs.

Context of the Standard
Version control is a tool that programmers use to 1) keep track of how a software program changes and develops over time, 2) preserve a record of changes made to a program in case it becomes necessary to restore older versions in the event of data loss, and 3) facilitate collaboration among many team members as they all work on the same piece of software. There are many kinds of version control tools, "git" being the most popular. Git keeps track of software as it changes, allowing programmers to revert to previous states if necessary. Additionally, it allows programmers to add in code to a common base independently without overwriting other team members' contributions. Tools like Git help programmers build software iteratively, adding and testing one element at a time.

Essential Skills	Essential Questions	Essential Vocabulary
Students should *demonstrate* these skills: • Create a program while using best practices of the program development cycle (design, code, test, debug). • Use proper version control methodologies to manage the iterative development process of a individual or collaborative program.	Students should *investigate* these concepts: • Why can't programmers use a tool like Google Docs to effectively collaborate on a code segment? • How might the development process be described as linear instead of cyclical?	Students should *apply* these terms in context: • Debugging • Incremental • Iterative • Program Development Cycle • Testing • Version Control

Impacts of Computing

CSF.22 The student will use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields.

Context of the Standard
Making software is a complex task that requires the help of many people with different areas of expertise. Programmers have technical knowledge about how to write code, but before they begin creating applications designers often create high-level, conceptual models to help guide the programming process. After programmers create a prototype, user experience analysts will test the code to see if it meets the needs of users and solves the problems the designers set out to address. Oftentimes, artists will create assets that fill out the application either by making the software easier to use or more pleasant to interact with. Software developers use a host of tools and techniques to manage this collaborative process. Version control software (CSF.21) plays a large role along with project management tools (at the time this framework was written, Trello, Monday, and Jira are all popular tools). Designers may also make use of a design or production methodology, a process that guides the development and implementation of ideas during software development. Popular frameworks (at the time this framework was written) include Agile, Lean, and Kanban. Digital tools and highly structured methodologies create a common language for designers and developers around the world. Development teams are often made of up of people from many different geographic locations collaborating asynchronously with a common goal in mind. However, software developers also use these tools to "outsource" certain tasks to people who will work for less money, especially people who live in countries which do not legally protect their workers from exploitation.

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills: • Design a simple program collaboratively using either a flowchart or pseudocode.	Students should *investigate* these concepts: • How could programmers work collaboratively on a project without being near each other?	Students should *apply* these terms in context: • Collaboration • Crowdsource • Flowchart

Essential Skills	Essential Questions	Essential Vocabulary
• Create a program collaboratively using paired programming or another group-focused programming tool.	• How can people in varied backgrounds and professions help identify mistakes in a program?	• Paired Programming • Prototype • Pseudocode

CSF.23 The student will evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.

Context of the Standard
Computing is pervasive in global society, fundamentally shaping economies, relationships, commerce, governance, and culture. There are many positive benefits to computing's impact, along with significant negative consequences. The digital economy is incredibly large, and technology skills are often prerequisites to employment in many fields. Large technology companies often invest in public education to create workers that will fill their expanding need for labor in creative and non-creative roles. Technology companies also mediate many social relationships in contemporary life; this allows people to learn many things using online resources, but also creates space for people to isolate themselves within often toxic or regressive ideologies. The data industry allows users to have access to powerful machine learning tools to help solve seemingly intractable problems. This access could also violate individuals' privacy and agency. In many ways, technology companies play outsized roles in shaping political and social discourse, having an unmeasured impact on cultural discourses within communities.

Essential Skills	Essential Questions	Essential Vocabulary
Students should *demonstrate* these skills: • Discuss the trade-off between technology use and personal privacy rights how that might impact an individual's agency.	Students should *investigate* these concepts: • Why is it important for technology creators/designers to think about the implications (both positive and negative) their innovations have on the way we live?	Students should *apply* these terms in context: • Economical Impacts • Ethical Impacts • Cultural Impacts • Personal Impacts

Essential Skills	Essential Questions	Essential Vocabulary
• Evaluate how a specific computing technology impacts society from ethical, global, and/or economic perspectives.		• Social Impacts • Trade-off

CSF.24 The student will explain the beneficial and harmful effects that intellectual property laws can have on innovation, including the impact of open source software.

Context of the Standard
Intellectual property laws are intended to protect creative work from unauthorized replication or reuse. When software is “open source,” it means that the developers have relinquished their exclusive intellectual property rights and have made their work publicly available for other programmers to use, alter, or reuse in different contexts. Intellectual property laws make it difficult to innovate because software developers cannot use the software in question to learn from or build on until the rights expire or the owners release the source code. On the other hand, it is difficult to make a living creating software without some measure of protection against people taking credit for work that is not their own. The software development industry has a complex relationship to intellectual property. Often, the product that technology companies are creating is not the software itself, but access to data from users. In these cases, software development companies are incentivized to release their source code so that more programmers can potentially work for the company in the future (Facebook is a good example). Software developers are also contributors to open source projects as an act of community-building. Open source software is an excellent resource for people who are learning to code, and it makes it easy for people to create tools that solve problems for users or other software developers. When software developers release open-source code, they will often specify a license which explains the terms under which the open source software may be used (e.g., MIT license).

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills: • Discuss the benefits and disadvantages of open source software. • Understand the purpose of intellectual property laws.	Students should *investigate* these concepts: • How does the ownership of intellectual property impact your work in creating programs and completing projects?	Students should *apply* these terms in context: • Intellectual Property Laws • Proprietary • Open Source Software

CSF.25 The student will explain the privacy concerns related to the collection and generation of data through automated processes that are not always evident to users.

Context of the Standard
Human-computer interaction generates massive amounts of data. In the contemporary digital economy, companies use this data to support the advertising industry, targeting customers with ads based on their Internet use and history. User data is a commodity that is traded among different companies who either purchase access to data, or raw data itself to use as they train machine learning algorithms to automate different services. Concerns related to privacy include identity theft, cyberstalking, catfishing, fraudulent purchases, etc. Inadvertent data collection makes many people feel as though their privacy is being invaded, particularly if that data is being collected without the user's knowledge. An example of inadvertent data collection can be seen with voice-activated devices. These devices have active microphones which can listen for key terms, but collect ambient audio as well. This audio can be used to gather other data that can be analyzed by developers, businesses, or organizations.

Essential Skills	**Essential Questions**	**Essential Vocabulary**
Students should *demonstrate* these skills:	Students should *investigate* these concepts:	Students should *apply* these terms in context:

Essential Skills	**Essential Questions**	**Essential Vocabulary**
• Investigate a well-known app and examine the data that is collected from users. • Explain how personal data might be used to make decisions that have positive and negative impacts on user groups.	• What exists in your digital footprint based on data collected from the apps and programs you use? • What are the best ways to protect your data privacy?	• Algorithm • Data Analysis • Data Collection • Privacy • User