2018 Mississippi College- and Career-Readiness Standards for Computer Science

2018 Mississippi College- and Career-Readiness Standards for Computer Science

Carey M. Wright, Ed.D., State Superintendent of Education

Kim S. Benton, Ed.D., Chief Academic Officer

Jean Massey, Executive Director, Office of Secondary Education

Nathan Oakley, Ph.D., Executive Director, Office of Elementary Education and Reading

Mike Mulvihill, Bureau Director, Office of Career and Technical Education

Important notes about Computer Science Standards

- The 2018 Computer Science Standards are in place to serve as guidance for districts moving forward with incorporating computer science content and courses into their schools.
- While these standards are not required to be taught in all schools at this time, they do provide the standards for implementation of computer science curricula and resources.
- All Computer Science courses currently in place, will be aligned to the new 2018 Computer Science Standards.
- Carnegie units earned in Computer Science meet the technology credit needed for graduation and admission to postsecondary schools in MS.

Mississippi Department of Education
Post Office Box 771
Jackson, Mississippi
39205-0771

Office of Elementary Education and Reading
Office of Secondary Education
Office of Career and Technical Education
601-359-2586
www.mde.k12.ms.us/ESE

The Mississippi State Board of Education, the Mississippi Department of Education, the Mississippi School for the Arts, the Mississippi School for the Blind, the Mississippi School for the Deaf, and the Mississippi School for Mathematics and Science do not discriminate on the basis of race, sex, color, religion, national origin, age, or disability in the provision of educational programs and services or employment opportunities and benefits. The following office has been designated to handle inquiries and complaints regarding the nondiscrimination policies of the above mentioned entities:

Director, Office of Human Resources
Mississippi Department of Education

Contents

Acknowledgements

The Mississippi Department of Education gratefully acknowledges the hard work of the following individuals for their involvement in developing the *Mississippi College- and Career-Readiness Standards for Computer Science* and the supporting documents.

COMPUTER SCIENCE STANDARDS REVIEW COMMITTEE MEMBERS (2017)

Melissa Banks	Mississippi Department of Education
Stacy Brooks	Mississippi Department of Education
Dane Conrad	Hattiesburg Public School District
Tracye Crane	Pontotoc City School District
Mary Dunaway	Rankin County School District
Eva Harvell	Pascagoula-Gautier School District
Kay Heath	Lowndes County School District
Randy Lynn	Maris, West & Baker Advertising/Kids Code Mississippi
Jennifer McCaskill	Rankin County School District
Alice Rainwater	Rankin County School District
Ross Randall	Lamar County School District
Donna Reese	Mississippi State University
Brandon Sesser	East Mississippi Community College, Mayhew
Shelley Songy	Lamar County School District
Kevin Walker	US Army Corp of Engineers, ERDC

COORDINATION AND EDITING (2017)

Sandra Gibson	Research and Curriculum Unit, Mississippi State University
Shelly Hollis	Research and Curriculum Unit, Mississippi State University
Jean Massey	Mississippi Department of Education
Mike Mulvihill	Mississippi Department of Education
Carl Smith	Research and Curriculum Unit, Mississippi State University

COMPUTER SCIENCE STEERING COMMITTEE (2016)

Kameron Ball	C Spire
Melissa Banks	Mississippi Department of Education
David Campbell	Vicksburg Warren Schools
Dane Conrad	Hattiesburg Public School District
Marla Davis	Mississippi Department of Education
Roberto Gallardo	Intelligent Community Institute
Kelley Gonzales	Rankin County School District
Gavin Guynes	Madison County School District
Eva Harvell	Pascagoula-Gautier School District
Kay Heath	Lowndes County School District
Jacqueline Jackson	Jackson State University
Alan Jones	C Spire
Randy Lynn	Maris, West & Baker Advertising/Kids Code Mississippi
Tim Martin	Clinton School District
Jennifer McCaskill	Rankin County School District
Brad Mixon	Booneville School District
Rachel Morgan	Tupelo School District
Mike Mulvihill	Mississippi Department of Education

COMPUTER SCIENCE STEERING COMMITTEE (2016), Continued

Lauren Olsen	Gulfport School District
Rodney Pearson	Mississippi State University
Debbie Raddin	MC STEM Institute
Alice Rainwater	Rankin County School District
Ross Randall	Lamar County School District
Bill Rayburn	FNC, Inc.
Donna Reese	Mississippi State University
Brandon Sesser	East Mississippi Community College, Mayhew
Chad Shealy	Vicksburg Warren School District
Subrimanian Gopinath	University of Southern Mississippi
Kevin Walker	US Army Corp of Engineers, ERDC
Vicki Webster	Delta State University
Maureen Whann	BancorpSouth

Various sets of standards and standards-related documents were used in the development of the Mississippi Computer Science Standards including:

- CSTA K-12 Computer Science Standards: http://www.csteachers.org/page/standards
- The K-12 Computer Science Framework: https://k12cs.org/
- Approved standards from the following states:
  - Massachusetts: http://www.doe.mass.edu/frameworks/dlcs.pdf
  - Wisconsin: https://dpi.wi.gov/sites/default/files/imce/computer-science/ComputerScienceStandardsFINALADOPTED.pdf

Attribution

The K-12 Computer Science Framework, led by the Association for Computing Machinery, Code.org, Computer Science Teachers Association, Cyber Innovation Center, and National Math and Science Initiative in partnership with states and districts, informed the development of this work.

The CSTA Standards Revision Task Force crafted standards by combining concept statements and practices from the Framework. The Task Force also used descriptive material from the Framework when writing examples and clarifying statements to accompany the standards.

For more information about the Framework, please visit k12cs.org.

Introduction

Mission Statement

The Mississippi Department of Education (MDE) is dedicated to student success, which includes improving student achievement in science, equipping citizens to solve complex problems, and establishing fluent communication skills within a technological environment. The Mississippi College- and Career-Readiness Standards provide a consistent, clear understanding of what students are expected to know and be able to do by the end of each grade level or course. The standards are designed to be robust and relevant to the real world, reflecting the knowledge and skills that students need for success in college and careers and allowing students to compete in the global economy.

Purpose

In an effort to closely align instruction for students who are progressing toward postsecondary study and the workforce, the *2018 Mississippi College- and Career-Readiness Standards (MS CCRS) for Computer Science* includes grade- and course-specific standards for K-12 computer science. Mississippi has adapted these standards from the nationally developed *Computer Science Teachers Association K-12 Computer Science Standards, Revised 2017*. (CSTA, 2017)

This document is designed to provide K-12 computer science teachers with a basis for curriculum development. In order to prepare students for careers and college, it outlines what knowledge students should obtain, and the types of skills students must master upon successful completion of each grade level. The *2018 MS CCRS for Computer Science* reflect national expectations while focusing on postsecondary success, but they are unique to Mississippi in addressing the needs of our students and teachers. The standards' content centers around seven practices: fostering an inclusive computing culture, collaborating around computing, recognizing and defining computational problems, developing and using abstractions, creating computational artifacts, testing and refining computational artifacts, and communicating about computing. Instruction in these areas is designed for a greater balance between content and process. Teachers are encouraged to transfer more ownership of the learning process to students, who can then direct their own learning and develop a deeper understanding of computer science and the problem-solving process. Doing so will produce students that will become more capable, independent, and scientifically literate adults.

Implementation

The *2018 MS CCRS for Computer Science* will be implemented during the 2018-2019 school year.

2018 Mississippi College- and Career-Readiness Standards for Computer Science Overview

Research and Background Information

The MDE sponsored research into the state of computer science education in the US beginning in late fall of 2015. Over the course of eight months, this research revealed an enormous shortage in computer scientists across the country and an almost nonexistent K-12 computer science pathway. The Bureau of Labor Statistics predicts a shortage of approximately 1 million computer science workers by 2024, and very few states offered any computer science courses or instruction at the K-12 level in 2015. A Google/Gallup report showed that while 9 out of 10 parents believed that it was important for their child to learn about computer science in school, only 1 in 4 schools across the nation offered any computer science courses (Google/Gallup, 2015). As of June 1, 2016, only five states had computer science standards. (Tilley-Coulson, 2016).

In response to this information, a steering committee was formed to design a plan of action to incorporate computer science education in Mississippi's K-12 schools. This committee was comprised of K-12 administrators, technology directors, and teachers, as well as postsecondary instructors and industry leaders. The committee began meeting in October 2015 and, over the course of the next several months, developed a recommendation to conduct a three-year pilot program during which time the state would explore curricula, teacher licensure, and standards development. Members of the steering committee and other Mississippi educators took part in the review of the national Computer Science Framework development in 2016 and submitted feedback to the writers. Once the framework was completed and the national Computer Science Teacher's Association's Computer Science Standards were revised, members of the steering committee and other Mississippi educators reviewed the standards and, with only minor editorial changes, recommended adopting these standards as a starting place for Mississippi Computer Science Standards. The document that follows is a result of those recommendations.

Computer science and the technologies it enables rest at the heart of our economy and the way we live our lives. To be well-educated citizens in a computing-intensive world and to be prepared for careers in the 21st century, our students must have a clear understanding of the principles and practices of computer science. The Mississippi Computer Science Standards delineate a core set of learning objectives designed to provide the foundation for a complete computer science curriculum and its implementation at the K-12 level. To this end, the Mississippi standards:

- Introduce the fundamental concepts of computer science to all students, beginning at the elementary school level
- Present computer science at the secondary school level in a way that can fulfill a computer science graduation credit
- Encourage additional secondary-level computer science courses that will allow interested students to study facets of computer science in more depth and prepare them for entry into the work force or college
- Increase the availability of rigorous computer science for all students, especially those who are members of underrepresented groups

The standards have been written by educators to be coherent and comprehensible to teachers, administrators, and policy makers. Levels 1A, 1B, 2, and 3A are the computer science standards for **ALL students**. The Level 3B standards are intended for students who wish to pursue the study of computer science in high school beyond what is required for all students (specialty or elective courses).

Connection to the K-12 Computer Science Framework

The *K-12 Computer Science Framework* (k12cs.org) provides overarching, high-level guidance per grade bands, while the standards provide detailed, measurable student performance expectations. The framework was considered as a primary input for the standards development process.

These standards were crafted by combining concept statements and practices from the framework. Descriptive material from the framework was also used when writing examples and clarifying statements to accompany the standards.

Concepts

The core concepts of the K-12 Computer Science Framework represent major content areas in the field of computer science. The core concepts are delineated by multiple subconcepts that represent specific ideas within each concept. The learning progressions for each subconcept provide a thread connecting student learning from kindergarten to 12th grade.

Core concepts of the framework:

- Computing systems
- Networks and the Internet
- Data and analysis
- Algorithms and programming
- Impacts of computing

Each core concept is described in an overview and delineated by multiple subconcepts that represent the specific ideas within that core concept. For example, the data and analysis core concept contains four subconcepts: collection, storage, visualization and transformation, and inference and models. Subconcept overviews are provided to describe the subconcepts and summarize how learning progresses across multiple grade bands.

Crosscutting concepts are themes that illustrate connections among different concept statements. They are integrated into concept statements, instead of existing as an independent dimension of the framework. The crosscutting concepts that are represented in each concept statement are noted in the statement's descriptive material.

Crosscutting concepts of the framework:

- Abstraction
- System relationships
- Human-computer interaction
- Privacy and security
- Communication and coordination

Abstraction: An abstraction is the result of reducing a process or set of information to a set of important characteristics for computational use. Abstractions establish interactions at a level of reduced complexity by managing the more complex details below the level of interaction. An abstraction can be created to generalize a range of situations by picking out common properties minus specific implementations details.

System relationships: The parts of a system are interdependent and organized for a common purpose. A systems perspective provides the opportunity to decompose complex problems into parts that are easier

to understand, develop, fix, and maintain. General systems principles include feedback, control, efficiency, modularity synthesis, emergence, and hierarchy.

Human-computer interaction: Humans interact directly with computers, including as laptops and smartphones, but also other devices, such as cars and home appliances, which have embedded computers. Developing effective and accessible user interfaces involves the integration of technical knowledge and social science and encompasses both designer and user perspectives.

Privacy and security: Privacy is the ability to seclude information and express it selectively. It includes controls for the collection, access, use, storage, sharing, and alteration of information. Security refers to the safeguards surrounding information systems and includes protection from theft or damage to hardware, software, and the information in the systems. Security supports privacy.

Communication and coordination: Processes in computing are characterized by the reliable exchange of information between autonomous agents (communication) that cooperate toward common outcomes that no agent could produce alone (coordination). Communication and coordination are distinct but not independent processes. What is special about computing is the scale at which communication and coordination work.

Core Concepts and Subconcepts Overview

Computing systems

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

Devices	Many everyday objects contain computational components that sense and act on the world. In early grades, students learn features and applications of common computing devices. As they progress, students learn about connected systems and how the interaction between humans and devices influences design decisions.
Hardware and software	Computing systems use hardware and software to communicate and process information in digital form. In early grades, students learn how systems use both hardware and software to represent and process information. As they progress, students gain a deeper understanding of the interaction between hardware and software at multiple levels within computing systems.
Troubleshooting	When computing systems do not work as intended, troubleshooting strategies help people solve the problem. In early grades, students learn that identifying the problem is the first step to fixing it. As they progress, students learn systematic problem-solving processes and how to develop their own troubleshooting strategies based on a deeper understanding of how computing systems work.

Networks and the Internet

Computing devices typically do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing. Networks and communication systems provide greater connectivity in the computing world by providing fast, secure communication and facilitating innovation.

Network communication and organization	Computing devices communicate with each other across networks to share information. In early grades, students learn that computers connect them to other people, places, and things around the world. As they progress, students gain a deeper understanding of how information is sent and received across different types of networks.
Cybersecurity	Transmitting information securely across networks requires appropriate protection. In early grades, students learn how to protect their personal information. As they progress, students learn increasingly complex ways to protect information sent across networks.

Data and analysis

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

Collection	Data is collected with both computational and noncomputational tools and processes. In early grades, students learn how data about themselves and their world is collected and used. As they progress, students learn the effects of collecting data with computational and automated tools.
Storage	Core functions of computers are storing, representing, and retrieving data. In early grades, students learn how data is stored on computers. As they progress, students learn how to evaluate different storage methods, including the tradeoffs associated with those methods.
Visualization and transformation	Data is transformed throughout the process of collection, digital representation, and analysis. In early grades, students learn how transformations can be used to simplify data. As they progress, students learn about more complex operations to discover patterns and trends and communicate them to others.
Inference and models	Data science is one example where computer science serves many fields. Computer science and science use data to make inferences, theories, or predictions based upon the data collected from users or simulations. In early grades, students learn about the use of data to make simple predictions. As they progress, students learn how models and simulations can be used to examine theories and

understand systems and how predictions and inferences are affected by more complex and larger data sets.

Algorithms and programming

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

Algorithms	Algorithms are designed to be carried out by both humans and computers. In early grades, students learn about age-appropriate algorithms from the real world. As they progress, students learn about the development, combination, and decomposition of algorithms, as well as the evaluation of competing algorithms.
Variables	Computer programs store and manipulate data using variables. In early grades, students learn that different types of data, such as words, numbers, or pictures, can be used in different ways. As they progress, students learn about variables and ways to organize large collections of data into data structures of increasing complexity.
Control	Control structures specify the order in which instructions are executed within an algorithm or program. In early grades, students learn about sequential execution and simple control structures. As they progress, students expand their understanding to combinations of structures that support complex execution.
Modularity	Modularity involves breaking down tasks into simpler tasks and combining simple tasks to create something more complex. In early grades, students learn that algorithms and programs can be designed by breaking tasks into smaller parts and recombining existing solutions. As they progress, students learn about recognizing patterns to make use of general, reusable solutions for commonly occurring scenarios and clearly describing tasks in ways that are widely usable.
Program development	Programs are developed through a design process that is often repeated until the programmer is satisfied with the solution. In early grades, students learn how and why people develop programs. As they progress, students learn about the tradeoffs in program design associated with complex decisions involving user constraints, efficiency, ethics, and testing.

Impacts of computing

Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and responsible person should understand the social implications of the digital world, including equity and access to computing.

Culture	Computing influences culture—including belief systems, language, relationships, technology, and institutions—and culture shapes how people engage with and access computing. In early grades, students learn how computing can be helpful and harmful. As they progress, students learn about tradeoffs associated with computing and potential future impacts of computing on global societies.
Social interactions	Computing can support new ways of connecting people, communicating information, and expressing ideas. In early grades, students learn that computing can connect people and support interpersonal communication. As they progress, students learn how the social nature of computing affects institutions and careers in various sectors.
Safety, law, and ethics	Legal and ethical considerations of using computing devices influence behaviors that can affect the safety and security of individuals. In early grades, students learn the fundamentals of digital citizenship and appropriate use of digital media. As they progress, students learn about the legal and ethical issues that shape computing practices.

Practices

The *K-12 Computer Science Framework's* practices are the behaviors that computationally literate students use to fully engage with the core concepts of computer science. Concepts and practices are integrated to provide complete experiences for students engaging in computer science. While the practices naturally integrate and overlap with one another, they are displayed in an order that suggests a process for developing computational artifacts. Four of the practices are also called out as aspects of computational thinking. There are seven core practices:

1. Fostering an inclusive computing culture
2. Collaborating around computing
3. Recognizing and defining computational problems
4. Developing and using abstractions
5. Creating computational artifacts
6. Testing and refining computational artifacts
7. Communicating about computing

The practices use a narrative to describe how students should exhibit each practice with increasing sophistication from kindergarten to grade 12. In addition to describing the progression, these narratives also provide some examples of the interrelatedness of the practice statements and the ways in which these statements build upon one another.

Computational thinking

Computational thinking is at the heart of the computer science practices and is delineated by practices 3-6. Practices 1, 2, and 7 are independent, general practices in computer science that complement computational thinking.

Computational thinking refers to the thought processes involved in expressing solutions as computational steps or algorithms that can be carried out by a computer (Cuny, Snyder, & Wing, 2010; Aho, 2011; Lee, 2016). This definition draws on the idea of formulating problems and solutions in a form that can be carried out by an information-processing agent (Cuny, Snyder, & Wing, 2010) and the idea that the solutions should take the specified form of computational steps and algorithms to be executed by a computer (Aho, 2011; Lee, 2016). Computational thinking requires understanding the capabilities of computers, formulating problems to be addressed by a computer, and designing algorithms that a computer can execute. The most effective context and approach for developing computational thinking is learning computer science; they are intrinsically connected.

Computational thinking is essentially a problem-solving process that involves designing solutions that capitalize on the power of computers; this process begins before a single line of code is written. Computers provide benefit in terms of memory, speed, and accuracy of execution. Computers also require people to express their thinking in a formal structure, such as a programming language. Similar to writing notes on a piece of paper to "get your thoughts down," creating a program allows people to externalize their thoughts in a form that can be manipulated and scrutinized. Programming allows students to think about their thinking; by debugging a program, students debug their own thinking (Papert, 1980).

Despite what the name implies, computational thinking is fundamentally a human ability. In other words, "humans process information; humans compute" (Wing, 2008, p. 3718). This nuance is the basis for "unplugged" approaches to computer science (i.e., teaching computer science without computers) and explains how computational thinking can apply beyond the borders of computer science to a variety of disciplines, such as science, technology, engineering, and mathematics (STEM), and also the arts and humanities (Bundy, 2007).

The description of computational thinking in the *K-12 Computer Science Framework* extends beyond the general use of computers or technology in education to include specific skills, such as designing algorithms, decomposing problems, and modeling phenomena. If computational thinking can take place without a computer, conversely, using a computer in class does not necessarily constitute computational thinking. For example, a student is not necessarily using computational thinking when he or she enters data into a spreadsheet and creates a chart. However, this action can include computational thinking if the student creates algorithms to automate the transformation of the data or to power an interactive data visualization.

A computational artifact must be distinguished by evaluating the process used to create it (i.e., computational thinking), in addition to the characteristics of the product itself. For example, the same digital animation may be the result of carefully constructing algorithms that control when characters move and how they interact or simply selecting characters and actions from a predesignated template. In this example, it is the process used to create the animation that defines whether it can be considered a computational artifact. The assessment of computational thinking can be improved by having students explain their decisions and development process (Brennan & Resnick, 2012).

Practice 1: Fostering an inclusive computing culture

Building an inclusive and diverse computing culture requires strategies for incorporating perspectives from people of different genders, ethnicities, and abilities. Incorporating these perspectives involves understanding the personal, ethical, social, economic, and cultural contexts in which people operate. Considering the needs of diverse users during the design process is essential to producing inclusive computational products.

By the end of grade 12, students should be able to:

1.1 Include the unique perspectives of others and reflect on one's own perspectives when designing and developing computational products
1.2 Address the needs of diverse end users during the design process to produce artifacts with broad accessibility and usability
1.3 Employ self- and peer-advocacy to address bias in interactions, product design, and development methods

Practice 2: Collaborating around computing

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

By the end of grade 12, students should be able to:

2.1 Identify complex, interdisciplinary, real-world problems that can be solved computationally
2.2 Create team norms, expectations, and equitable workloads to increase efficiency and effectiveness
2.3 Solicit and incorporate feedback from, and provide constructive feedback to, team members and other stakeholders
2.4 Evaluate and select technological tools that can be used to collaborate on a project

Practice 3: Recognizing and defining computational problems

The ability to recognize appropriate and worthwhile opportunities to apply computation is a skill that develops over time and is central to computing. Solving a problem with a computational approach requires defining the problem, breaking it down into parts, and evaluating each part to determine whether a computational solution is appropriate.

By the end of grade 12, students should be able to:

3.1 Identify complex, interdisciplinary, real-world problems that can be solved computationally
3.2 Decompose complex real-world problems into manageable subproblems that could integrate existing solutions or procedures
3.3 Evaluate whether it is appropriate and feasible to solve a problem computationally

Practice 4: Developing and using abstractions

Abstractions are formed by identifying patterns and extracting common features from specific examples to create generalizations. Using generalized solutions and parts of solutions designed for broad reuse simplifies the development process by managing complexity.

By the end of grade 12, students should be able to:

- 4.1 Extract common features from a set of interrelated processes or complex phenomena
- 4.2 Evaluate existing technological functionalities and incorporate them into new designs
- 4.3 Create modules and develop points of interaction that can apply to multiple situations and reduce complexity
- 4.4 Model phenomena and processes and simulate systems to understand and evaluate potential outcomes

Practice 5: Creating computational artifacts

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

By the end of grade 12, students should be able to:

- 5.1 Plan the development of a computational artifact using an iterative process that includes reflection on and modification of the plan, taking into account key features, time and resource constraints, and user expectations
- 5.2 Create a computational artifact for practical intent, personal expression, or to address a societal issue
- 5.3 Modify an existing artifact to improve or customize it

Practice 6: Testing and refining computational artifacts

Testing and refinement is the deliberate and iterative process of improving a computational artifact. This process includes debugging (identifying and fixing errors) and comparing actual outcomes to intended outcomes. Students also respond to the changing needs and expectations of end users and improve the performance, reliability, usability, and accessibility of artifacts.

By the end of grade 12, students should be able to:

- 6.1 Systematically test computational artifacts by considering all scenarios and using test cases
- 6.2 Identify and fix errors using a systematic process
- 6.3 Evaluate and refine a computational artifact multiple times to enhance its performance, reliability, usability, and accessibility

Practice 7: Communicating about computing

Communication involves personal expression and exchanging ideas with others. In computer science, students communicate with diverse audiences about the use and effects of computation and the appropriateness of computational choices. Students write clear comments, document their work, and communicate their ideas through multiple forms of media. Clear communication includes using precise language and carefully considering possible audiences.

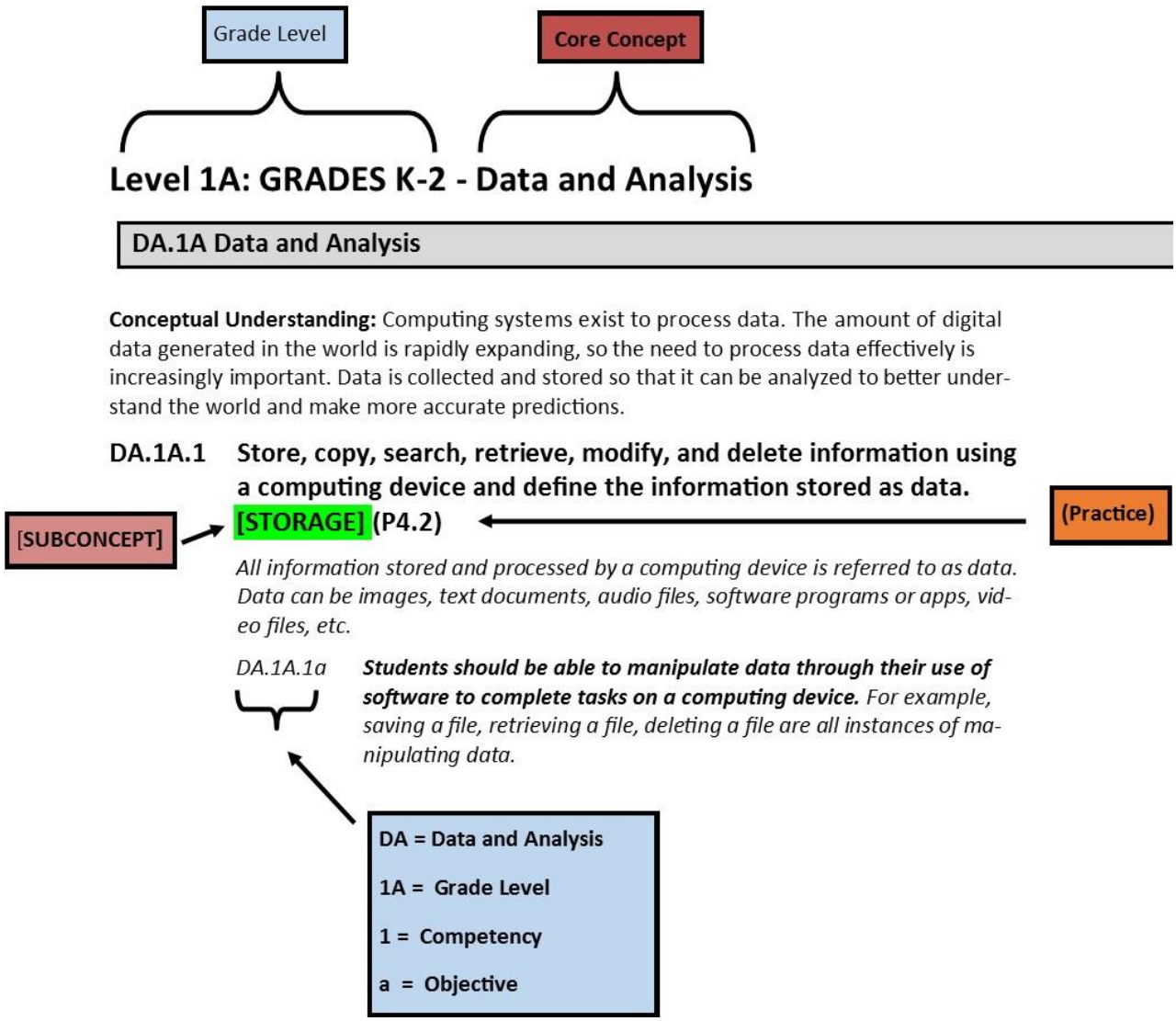
By the end of grade 12, students should be able to:

7.1 Select, organize, and interpret large data sets from multiple sources to support a claim

7.2 Describe, justify, and document computational processes and solutions using appropriate terminology consistent with the intended audience and purpose

7.3 Articulate ideas responsibly by observing intellectual property rights and giving appropriate attribution

Understanding the Computer Science Standards Format



Core concepts	Practices
1. Computing systems 2. Networks and the Internet 3. Data and analysis 4. Algorithms and programming 5. Impacts of computing	1. Fostering an inclusive computing culture 2. Collaborating around computing 3. Recognizing and defining computational problems 4. Developing and using abstractions 5. Creating computational artifacts 6. Testing and refining computational artifacts 7. Communicating about computing

Level 1A: GRADES K-2

Level 1A: GRADES K-2 - Computing Systems

Computing Systems (CS.1A)

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

CS.1A.1 **Select and operate appropriate software to perform a variety of tasks, and recognize that users have different needs and preferences for the technology they use. [DEVICES] (P1.1)**

CS.1A.1a ***Students should be able to select the appropriate app/program to use for tasks they are required to complete.*** *For example, if students are asked to draw a picture, they should be able to open and use a drawing app/program to complete this task, or if they are asked to create a presentation, they should be able to open and use presentation software.*

CS.1A.1b ***With teacher guidance, students should compare and discuss preferences for software with the same primary functionality.*** *Students could compare different web browsers or word processing, presentation, or drawing programs*

CS.1A.2 **Use appropriate terminology in identifying and describing the function of common physical components of computing systems (hardware). [HARDWARE & SOFTWARE] (P7.2)**

A computing system is composed of hardware and software. Hardware consists of physical components.

CS.1A.2a ***Students should be able to identify and describe the function of external hardware, such as desktop computers, laptop computers, tablet devices, monitors, keyboards, mice, and printers.***

CS.1A.3 **Describe basic hardware and software problems using accurate terminology. [TROUBLESHOOTING] (P6.2, P7.2)**

CS.1A.3a ***Students should be able to communicate a problem with accurate terminology (e.g., when an app or program is not working as expected, a device will not turn on, the sound does not work, etc.).*** *Ideally, students would be able to use simple troubleshooting strategies, including turning a device off and on to reboot it, closing and reopening an app, turning on speakers, or plugging in headphones. These are, however, not specified in the standard, because these problems may not occur.*

Level 1A: GRADES K-2 - Networks and the Internet

Networks and the Internet (NI.1A)

Conceptual understanding: Computing devices typically do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing. Networks and communication systems provide greater connectivity in the computing world by providing fast, secure communication and facilitating innovation.

NI.1A.1 **Explain what passwords are and why we use them. [CYBERSECURITY] (P7.3)**
Learning to protect one's device or information from unwanted use by others is an essential first step in learning about cybersecurity. They should appropriately use and protect the passwords they are required to use.

NI.1A.2 **Students should understand that computers connect them to people, places, and things around the world. [NETWORK COMMUNICATION & ORGANIZATION] (P7.3)**

Level 1A: GRADES K-2 - Data and Analysis

Data and Analysis (DA.1A)

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

DA.1A.1 **Store, copy, search, retrieve, modify, and delete information using a computing device and define the information stored as data. [STORAGE] (P4.2)**
All information stored and processed by a computing device is referred to as data. Data can be images, text documents, audio files, software programs or apps, video files, etc.

DA.1A.1a ***Students should be able to manipulate data through their use of software to complete tasks on a computing device.*** *For example, saving a file, retrieving a file, deleting a file are all instances of manipulating data.*

DA.1A.2 **Collect and present the same data in various visual formats. [COLLECTION, VISUALIZATION, & TRANSFORMATION] (P7.1, P4.4)**
The collection and use of data about the world around them is a routine part of life and influences how people live.

DA.1A.2a ***Students should be able to collect data.*** *For example, students could collect data on the weather, such as sunny days versus rainy days, the temperature at the beginning of the school day and end of the school day, or the inches of rain over the course of a storm. Students could count the number of pieces of each color of candy in a bag of candy, such as Skittles or M&Ms. Students could create surveys of things that interest them, such as favorite foods, pets, or TV shows, and collect answers to their surveys from their peers and others.*

DA.1A.2b ***Students should be able to present data in various visual formats.*** *For example, the data collected could be organized into two or more visualizations, such as a bar graph, pie chart, or pictograph.*

DA.1A.3 **Identify and describe patterns in data visualizations, such as charts or graphs, to make predictions. [INFERENCEE & MODELS] (P4.1)**
All data can be used to make inferences or predictions about the world.

DA.1A.3a ***Students should be able to analyze data in visual formats.*** *For example, students could analyze a graph or pie chart of the colors in a bag of candy or the averages for colors in multiple bags of candy. Students could analyze graphs of temperatures taken at the beginning of the school day and end of the school day.*

DA.1A.3b ***Students should be able to identify patterns and make predictions based on the patterns.*** *For example, students could identify the patterns for which colors are most and least represented in bags of candy, and then make a prediction as to which colors will have most and least in a new bag of candy. Students could*

identify the patterns of when temperatures rise and fall, and predict if they think the temperature will rise or fall at a particular time of the day, based on the pattern observed.

Level 1A: GRADES K-2 - Algorithms and Programming

Algorithms and Programming (AP.1A)

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

AP.1A.1 Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks. [ALGORITHMS] (P4.4)

Composition is the combination of smaller tasks into more complex tasks.

AP.1A.1a ***Students should be able to create and follow algorithms****. For example, students could create and follow algorithms for making simple foods, brushing their teeth, getting ready for school, participating in cleanup time. Students may demonstrate understanding visually, orally, or in writing.*

AP.1A.2 Model the way programs store and manipulate data by using numbers or other symbols to represent information. [VARIABLES] (P4.4)

Information in the real world can be represented in computer programs.

AP.1A.2a ***Students should be able to model data storage and manipulation by using representative symbols.*** *For example, students could use thumbs up/down as representations of yes/no, use arrows when writing algorithms to represent direction, or encode and decode words using numbers, pictographs, or other symbols to represent letters or words.*

AP.1A.3 Develop programs with sequences and simple loops to express ideas or address a problem. [CONTROL] (P5.2)

Programming is used as a tool to create products that reflect a wide range of interests. Control structures specify the order in which instructions are executed within a program. Sequences are the order of instructions in a program. For example, if dialogue is not sequenced correctly when programming a simple animated story, the story will not make sense. If the commands to program a robot are not in the correct order, the robot will not complete the task desired. Loops allow for the repetition of a sequence of code multiple times. For example, in a program to show the life cycle of a butterfly, a loop could be combined with move commands to allow continual but controlled movement of the character.

AP.1A.3a ***Students should be able to express ideas or address problems by developing programs with sequences and simple loops.***

AP.1A.4 Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions. [MODULARITY] (P3.2)

Decomposition is the act of breaking down tasks into simpler tasks.

AP.1A.4a ***Students should be able to break down the steps needed to solve a problem into a precise sequence of instructions.*** *For example, students could break down the steps needed to make a peanut butter and jelly sandwich, to brush their teeth, to draw a shape, to move a character across the screen, or to solve a level of a coding app. Students may demonstrate understanding visually, orally, or in*

writing.

AP.1A.5 Develop plans that describe a program's sequence of events, goals, and expected outcomes. [PROGRAM DEVELOPMENT] (P5.1, P7.2)

Creating a plan for what a program will do clarifies the steps that will be needed to create a program and can be used to check if a program is correct.

AP.1A.5a ***Students should be able to develop and visually illustrate the plan for what a program will do.*** *For example, students could create a planning document, such as a story map, a storyboard, or a sequential graphic organizer, to illustrate what their program will do. Students at this stage may complete the planning process with help from their teachers.*

AP.1A.6 Give attribution when using the ideas and creations of others while developing programs. [PROGRAM DEVELOPMENT] (P7.3)

Using computers comes with a level of responsibility.

AP.1A.6a ***Students should credit artifacts that were created by others, such as pictures, music, and code.*** *Credit could be given orally, if presenting their work to the class, or in writing, if sharing work on a class blog or website. Proper attribution at this stage does not require a formal citation, such as in a bibliography or works cited document.*

AP.1A.7 Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops. [PROGRAM DEVELOPMENT] (P6.2)

Algorithms or programs may not always work correctly.

AP.1A.7a ***Students should be able to use various strategies, such as changing the sequence of the steps, following the algorithm in a step-by-step manner, or trial and error to fix problems in algorithms and programs.***

AP.1A.8 Using correct terminology, describe steps taken and choices made during the iterative process of program development. [PROGRAM DEVELOPMENT] (P7.2)

AP.1A.8a ***Students should be able to talk or write about the goals and expected outcomes of the programs they create and the choices that they made when creating programs.*** *This could be done using coding journals, discussions with a teacher, class presentations, or blogs.*

Level 1A: GRADES K-2 - Impacts of Computing

Impacts of Computing (IC.1A)

Conceptual understanding: Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and responsible person should understand the social implications of the digital world, including equity and access to computing.

IC.1A.1 Compare how people live and work before and after the implementation or adoption of new computing technology. [CULTURE] (P7.0)

Computing technology has positively and negatively changed the way people live and work. In the past, if students wanted to read about a topic, they needed access to a library to find a book about it. Today, students can view and read information on the Internet about a topic or they can download e-books about it directly to a device. Such information may be available in more than one language and could be read to a student, allowing for great accessibility.

IC.1A.1a ***Students should be able to compare how people live and work before and after the implementation or adoption of new computing technology.***

IC.1A.2 Work respectfully and responsibly with others online. [SOCIAL INTERACTIONS] (P2.1)

Online communication facilitates positive interactions, such as sharing ideas with many people, but the public and anonymous nature of online communication also allows intimidating and inappropriate behavior in the form of cyberbullying.

IC.1A.2a ***Students should demonstrate understanding of how to work with others online in a respectful and responsibly way.*** *Students could share their work on blogs or in other collaborative spaces online, taking care to avoid sharing information that is inappropriate or that could personally identify them to others.*

IC.1A.2a ***Students should be able to identify cyberbullying.*** *Students could provide feedback to others on their work in a kind and respectful manner and could tell an adult if others are sharing things they should not share or are treating others in an unkind or disrespectful manner on online collaborative spaces.*

IC.1A.3 Keep login information private and log off of devices appropriately. [SAFETY, LAW, & ETHICS] (P7.3)

People use computing technology in ways that can help or hurt themselves or others. Harmful behaviors, such as sharing private information and leaving public devices logged in, should be recognized and avoided.

IC.1A.3a ***Students should understand that some things like login details, their address, and other personally identifying information is private (secret).***

IC.1A.3b ***Students should know to always log off properly on any device used.***

Level 1B: GRADES 3-5

Level 1B: GRADES 3-5 - Computing Systems

Computing Systems (CS.1B)

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

CS.1B.1 Describe how internal and external parts of computing devices function to form a system. [DEVICES] (P7.2)

Computing devices often depend on other devices or components. For example, a robot depends on a physically attached light sensor to detect changes in brightness, whereas the light sensor depends on the robot for power. Keyboard input or a mouse click could cause an action to happen or information to be displayed on a screen; this could only happen because the computer has a processor to evaluate what is happening externally and produce corresponding responses.

CS.1B.1a ***Students should describe how devices and components interact using correct terminology.***

CS.1B.2 Model how computer hardware and software work together as a system to accomplish tasks. [HARDWARE & SOFTWARE] (P4.4)

In order for a person to accomplish tasks with a computer, both hardware and software are needed. At this stage, a model should only include the basic elements of a computer system, such as input, output, processor, sensors, and storage.

CS.1B.2a ***Students should model how computer hardware and software work together to accomplish tasks.*** *Students could draw a model on paper or in a drawing program, program an animation to demonstrate it, or demonstrate it by acting this out in some way.*

CS.1B.3 Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies. [TROUBLESHOOTING] (P6.2)

Although computing systems may vary, common troubleshooting strategies can be used on all of them.

CS.1B.3a ***Students should be able to identify common hardware and software problems.*** *Types of problems students might encounter include the device not responding, no power, no network, app crashing, no sound, or password entry not working.*

CS.1B.3b ***Students should identify and implement various troubleshooting strategies.*** *Such strategies may include rebooting the device, checking for power, checking network availability, closing and reopening an app, making sure speakers are turned on or headphones are plugged in, and making sure that the caps lock key is not on, to solve these problems, when possible.*

Level 1B: GRADES 3-5 - Networks and the Internet

Networks and the Internet (NI.1B)

Conceptual understanding: Computing devices typically do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing.

Networks and communication systems provide greater connectivity in the computing world by providing fast, secure communication and facilitating innovation.

NI.1B.1 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. [NETWORK COMMUNICATION & ORGANIZATION] (P4.4)

Information is sent and received over physical or wireless paths. It is broken down into smaller pieces called packets, which are sent independently and reassembled at the destination.

NI.1B.1a ***Students should demonstrate their understanding of how information flows over networks and the Internet.*** *This could be accomplished, for instance, by drawing a model of the way packets are transmitted, programming an animation to show how packets are transmitted, or demonstrating this through an unplugged activity which has them act it out in some way.*

NI.1B.2 Discuss real-world cybersecurity problems and how personal information can be protected. [CYBERSECURITY] (P3.1, 7.3)

Just as we protect our personal property offline, we also need to protect our devices and the information stored on them. Information can be protected using various security measures. These measures can be physical and/or digital.

NI.1B.2a **Students should be able to explain what passwords are and why we use them, and use strong passwords to protect devices and information from unauthorized access.**

Learning to protect one's device or information from unwanted use by others is an essential first step in learning about cybersecurity. Students are not required to use multiple strong passwords. They should appropriately use and protect the passwords they are required to use.

NI.1B.2b ***Students should be able to list several real-world cybersecurity issues and discuss how personal information can be protected.*** *Students could discuss or use a journaling or blogging activity to explain, orally or in writing, topics that relate to personal cybersecurity issues. Discussion topics could be based on current events related to cybersecurity or topics that are applicable to students, such as the necessity of backing up data to guard against loss, how to create strong passwords and the importance of not sharing passwords, or why we should install and keep antivirus software updated to protect data and systems.*

Level 1B: GRADES 3-5 - Data and Analysis

Data and Analysis (DA.1B)

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

DA.1B.1 Organize and present collected data visually to highlight relationships and support a claim. [COLLECTION, VISUALIZATION, & TRANSFORMATION] (P7.1)

Raw data has little meaning on its own. Data is often sorted or grouped to provide additional clarity. Organizing data can make interpreting and communicating it to others easier. Data points can be clustered by a number of commonalities. The same data could be manipulated in different ways to emphasize particular aspects or parts of the data set. For example, a data set

of sports teams could be sorted by wins, points scored, or points allowed, and a data set of weather information could be sorted by high temperatures, low temperatures, or precipitation.

DA.1B.1a ***Students should be able to collect data and present the information in an organized way to highlight relationships and support a claim.***

DA.1B.2 Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea. [INFERENCE & MODELS] (P7.1)

The accuracy of data analysis is related to how realistically data is represented. Inferences or predictions based on data are less likely to be accurate if the data is not sufficient or if the data is incorrect in some way.

DA.1B.2a ***Students should be able to refer to data to highlight or propose cause-and-effect relationships and predict outcomes when communicating an idea.*** *For example, in order to explore the relationship between speed, time, and distance, students could operate a robot at uniform speed and at increasing time intervals to predict how far the robot travels at that speed. In order to make an accurate prediction, one or two attempts of differing times would not be enough. The robot may also collect temperature data from a sensor, but that data would not be relevant for the task. Students must also make accurate measurements of the distance the robot travels in order to develop a valid prediction. Students could record the temperature at noon each day as a basis to show that temperatures are higher in certain months of the year. If temperatures are not recorded on nonschool days or are recorded incorrectly or at different times of the day, the data would be incomplete and the ideas being communicated could be inaccurate. Students may also record the day of the week on which the data was collected, but this would have no relevance to whether temperatures are higher or lower. In order to have sufficient and accurate data on which to communicate the idea, students might want to use data provided by a governmental weather agency.*

DA.1B.3 Store, copy, search, retrieve, modify, and delete information using a computing device and define the information stored as data. [STORAGE] (P4.2)

All information stored and processed by a computing device is referred to as data. Data can be images, text documents, audio files, software programs or apps, video files, etc.

DA.1B.3a ***Students should be able to manipulate data through their use of software to complete tasks on a computing device.*** *For example, saving, retrieving, and deleting files are all instances of manipulating data.*

Level 1B: GRADES 3-5 - Algorithms and Programming

Algorithms and Programming (AP.1B)

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

AP.1B.1 Compare and refine multiple algorithms for the same task and determine which is the most appropriate. [ALGORITHMS] (P6.3, P3.3)

Different algorithms can achieve the same result, though sometimes one algorithm might be

most appropriate for a specific situation.

AP.1B.1a ***Students should be able to look at different ways to solve the same task and decide which would be the best solution.*** *For example, students could use a map and plan multiple algorithms to get from one point to another. They could look at routes suggested by mapping software and change the route to something that would be better based on which route is shortest or fastest or would avoid a problem. Students might compare algorithms that describe how to get ready for school. Another example might be to write different algorithms to draw a regular polygon and determine which algorithm would be the easiest to modify or repurpose to draw a different polygon.*

AP.1B.2 Create programs that use variables to store and modify data. [VARIABLES] (P5.2)

Variables are used to store and modify data.

AP.1B.2a ***Students should understand how to use variables to store and modify data.*** *For example, students may use mathematical operations to add to the score of a game or subtract from the number of lives available in a game. The use of a variable as a countdown timer is another example.*

AP.1B.3 Create programs that include sequences, events, loops, and conditionals. [CONTROL] (P5.2)

Control structures specify the order (sequence) in which instructions are executed within a program and can be combined to support the creation of more complex programs. Events allow portions of a program to run based on a specific action.

AP.1B.3a ***Students should be able to create programs that include sequences, events, loops, and conditionals.*** *For example, students could write a program to explain the water cycle. When a specific component is clicked (event), the program would show information about that part of the water cycle. Conditionals allow for the execution of a portion of code in a program when a certain condition is true. For example, students could write a math game that asks multiplication fact questions and then uses a conditional to check whether or not the answer that was entered is correct. Loops allow for the repetition of a sequence of code multiple times. For example, in a program that produces an animation about a famous historical character, students could use a loop to have the character walk across the screen as they introduce themselves.*

AP.1B.4 Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process. [MODULARITY] (P3.2)

Decomposition is the act of breaking down tasks into simpler tasks.

AP.1B.4a ***Students should be able to break down problems into smaller, simpler tasks.*** *For example, students could create an animation by separating a story into different scenes. For each scene, they would select a background, place characters, and program actions.*

AP.1B.5 Modify, remix, or incorporate portions of an existing program into one's own work to develop something new or add more advanced features. [MODULARITY] (P5.3)

Programs can be broken down into smaller parts, which can be incorporated into new or existing programs.

AP.1B.5a ***Students should be able to modify and/or reuse portions of an existing program into their own work to create something new.*** *For example, students could modify prewritten code from a single-player game to create a two-player game with slightly different rules, remix and add another scene to an animated story, use code to make a ball bounce from another program in a new basketball game, or modify an image created by another student.*

AP.1B.6 Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences. [PROGRAM DEVELOPMENT] (P1.1, P5.1)

Planning is an important part of the iterative process of program development.

AP.1B.6a ***Students outline key features, time and resource constraints, and user expectations.***

AP.1B.6b ***Students should document the plan as, for example, a storyboard, flowchart, pseudocode, or story map.***

AP.1B.7 Observe intellectual property rights and give appropriate attribution when creating or remixing programs. [PROGRAM DEVELOPMENT] (P7.3)

Intellectual property rights can vary by country, but copyright laws give the creator of a work a set of rights that prevents others from copying the work and using it in ways that they may not like.

AP.1B.7a ***Students should identify instances of remixing, when ideas are borrowed and iterated upon, and credit the original creator.***

AP.1B.7b ***Students should also consider common licenses that place limitations or restrictions on the use of computational artifacts, such as images and music downloaded from the Internet.*** *At this stage, attribution should be written in the format required by the teacher and should always be included on any programs shared online.*

AP.1B.8 Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended. [PROGRAM DEVELOPMENT] (P6.1, P6.2)

As students develop programs, they should continuously test those programs to see that they do what was expected and fix (debug), any errors.

AP.1B.8a ***Students should be able to identify and debug simple errors in programs they create and in programs created by others.***

AP.1B.9 Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development. [PROGRAM DEVELOPMENT] (P2.2)

Collaborative computing is the process of performing a computational task by working in pairs or on teams. Because it involves asking for the contributions and feedback of others, effective collaboration can lead to better outcomes than working independently.

AP.1B.9a ***Students should take turns in different roles during program development, such as note taker, facilitator, program tester, or "driver" of the computer.***

AP.1B.10 Describe choices made during program development using code comments, presentations, and demonstrations. [PROGRAM DEVELOPMENT] (P7.2)

People communicate about their code to help others understand and use their programs. Another purpose of communicating one's design choices is to show an understanding of one's work.

AP.1B.10a ***Students should explain code choices using comments within the code, presentations, and demonstrations.*** *These explanations could manifest themselves as in-line code comments for collaborators and assessors or as part of a summative presentation, such as a code walk-through or coding journal.*

Level 1B: GRADES 3-5 - Impacts of Computing

Impacts of Computing (IC.1B)

Conceptual Understanding: Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and responsible person should understand the social implications of the digital world, including equity and access to computing.

IC.1B.1 **Discuss computing technologies that have changed the world and express how those technologies influence and are influenced by cultural practices. [CULTURE] (P7.1)**

New computing technology is created and existing technologies are modified for many reasons, including to increase their benefits, decrease their risks, and meet societal needs.

IC.1B.1a ***Students, with guidance from their teacher, should discuss topics that relate to the history of technology and the changes in the world due to technology.*** *Topics could be based on current news content, such as robotics, wireless Internet, mobile computing devices, GPS systems, wearable computing, or how social media has influenced social and political changes.*

IC.1B.2 **Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users. [CULTURE] (P1.2)**

The development and modification of computing technology are driven by people's needs and wants and can affect groups differently. Anticipating the needs and wants of diverse end users requires students to purposefully consider potential perspectives of users with different backgrounds, ability levels, points of view, and disabilities.

IC.1B.2a ***Students will demonstrate an understanding of diversity in ability and interests by developing artifacts and tools that use different methods of communication and/or appeal to different users.*** *For example, students may consider using both speech and text when they wish to convey information in a game. They may also wish to vary the types of programs they create, knowing that not everyone shares their own tastes.*

IC.1B.3 **Seek diverse perspectives for the purpose of improving computational artifacts. [SOCIAL INTERACTIONS] (P1.1)**

Computing provides the possibility for collaboration and sharing of ideas and allows the benefit of diverse perspectives.

IC.1B.3a ***Students will collaborate and receive feedback from others.*** *For example, students could seek feedback from other groups in their class or students at another grade level. Or, with guidance from their teacher, they could use video conferencing tools or other online collaborative spaces, such as blogs, wikis, forums, or website comments, to gather feedback from individuals and groups about programming projects.*

IC.1B.4 **Use public domain or creative commons media and refrain from copying or using material created by others without permission. [SAFETY, LAW, & ETHICS] (P7.3)**

Ethical complications arise from the opportunities provided by computing. The ease of sending and receiving copies of media on the Internet, such as video, photos, and music, creates the opportunity for unauthorized use, such as online piracy, and disregard of copyrights.

IC.1B.4a ***Students should consider the licenses on computational artifacts that they wish to use.*** *For example, the license on a downloaded image or audio file may have restrictions that prohibit modification, require attribution, or prohibit use entirely.*

Level 2: GRADES 6-8

Level 2: GRADES 6-8 - Computing Systems

Computing Systems (CS.2)

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

CS.2.1 **Recommend improvements to the design of computing devices based on an analysis of how users interact with the devices. [DEVICES] (P3.3)**

The study of human-computer interaction (HCI) can improve the design of devices, including both hardware and software.

CS.2.1a ***Students should make recommendations for existing devices (e.g., a laptop, phone, or tablet) or design their own components or interface (e.g., create their own controllers).*** *Teachers can guide students to consider usability through several lenses, including accessibility, ergonomics, and learnability. For example, assistive devices provide capabilities such as scanning written information and converting it to speech.*

CS.2.2 **Design projects that combine hardware and software components to collect and exchange data. [HARDWARE & SOFTWARE] (P5.1)**

Collecting and exchanging data involves input, output, storage, and processing. When possible, students should select the hardware and software components for their project designs by considering factors such as functionality, cost, size, speed, accessibility, and aesthetics.

CS.2.2a ***Students will design projects that use both hardware and software to collect and exchange data.*** *For example, components for a mobile app could include accelerometer, GPS, and speech recognition. The choice of a device that connects wirelessly through a Bluetooth connection versus a physical USB connection involves a tradeoff between mobility and the need for an additional power source for the wireless device.*

CS.2.3 **Systematically identify and fix problems with computing devices and their components. [TROUBLESHOOTING] (P6.2)**

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.

CS.2.3a ***Students will use a structured process to troubleshoot problems with computing systems and ensure that potential solutions are not overlooked.*** *Examples of troubleshooting strategies include following a troubleshooting flow diagram, making changes to software to see if hardware will work, checking connections and settings, and swapping in working components.*

Level 2: GRADES 6-8 - Networks and the Internet

Networks and the Internet (NI.2)

Conceptual Understanding: Computing devices typically do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing. Networks and communication systems provide greater connectivity in the computing world by providing

fast, secure communication and facilitating innovation.

NI.2.1 Model the role of protocols in transmitting data across networks and the Internet. [NETWORK COMMUNICATION & ORGANIZATION] (P4.4)

Protocols are rules that define how messages between computers are sent. They determine how quickly and securely information is transmitted across networks and the Internet, as well as how to handle errors in transmission.

NI.2.1a ***Students should model how data is sent using protocols to choose the fastest path, to deal with missing information, and to deliver sensitive data securely.*** *For example, students could devise a plan for resending lost information or for interpreting a picture that has missing pieces. The priority at this grade level is understanding the purpose of protocols and how they enable secure and errorless communication. Knowledge of the details of how specific protocols work is not expected.*

NI.2.2 Explain how physical and digital security measures protect electronic information. [CYBERSECURITY] (P7.2)

Information that is stored online is vulnerable to unwanted access. Examples of physical security measures to protect data include keeping passwords hidden, locking doors, making backup copies on external storage devices, and erasing a storage device before it is reused. Examples of digital security measures include secure router admin passwords, firewalls that limit access to private networks, and the use of a protocol, such as HTTPS, to ensure secure data transmission.

NI.2.2a ***Students will explain how physical and digital security measures protect electronic information.***

NI.2.3 Apply multiple methods of encryption to model the secure transmission of information. [CYBERSECURITY] (P4.4)

Encryption can be as simple as letter substitution or as complicated as modern methods used to secure networks and the Internet.

NI.2.3a ***Students should encode and decode messages using a variety of encryption methods, and they should understand the different levels of complexity used to hide or secure information.*** *For example, students could secure messages using methods like Caesar cyphers or steganography (i.e., hiding messages inside a picture or other data). They can also model more complicated methods, such as public key encryption, through unplugged activities.*

Level 2: GRADES 6-8 - Data and Analysis

Data and Analysis (DA.2)

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

DA.2.1 Represent data using multiple encoding schemes. [STORAGE] (P4.0)

Data representations occur at multiple levels of abstraction, from the physical storage of bits to the arrangement of information into organized formats (e.g., tables).

DA.2.1a ***Students should represent the same data in multiple ways.*** *For example, students could represent the same color using binary, RGB values, hex codes (low-level representations), as well as forms understandable by people, including words, symbols, and digital displays of the color (high-level representations).*

DA.2.2 **Collect data using computational tools and transform the data to make it more useful and reliable. [COLLECTION, VISUALIZATION, & TRANSFORMATION] (P6.3)**

As students continue to build on their ability to organize and present data visually to support a claim, they will need to understand when and how to transform data for this purpose.

DA.2.2a ***Students should transform data to remove errors, highlight or expose relationships, and/or make it easier for computers to process.*** *The cleaning of data is an important transformation for ensuring consistent format and reducing noise and errors (e.g., removing irrelevant responses in a survey). An example of a transformation that highlights a relationship is representing males and females as percentages of a whole instead of as individual counts.*

DA.2.3 **Refine computational models based on the data they have generated. [INFERENCE & MODELS] (P5.3, P4.4)**

A model may be a programmed simulation of events or a representation of how various data is related.

DA.2.3a ***Students will refine computational models by considering which data points are relevant, how data points relate to each other, and if the data is accurate.*** *For example, students may make a prediction about how far a ball will travel based on a table of data related to the height and angle of a track. The students could then test and refine their model by comparing predicted versus actual results and considering whether other factors are relevant (e.g., size and mass of the ball). Additionally, students could refine game mechanics based on test outcomes in order to make the game more balanced or fair.*

Level 2: GRADES 6-8 - Algorithms and Programming

Algorithms and Programming (AP.2)

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

AP.2.1 **Use flowcharts and/or pseudocode to address complex problems as algorithms. [ALGORITHMS] (P4.4, P4.1)**

Complex problems are problems that would be difficult for students to solve computationally.

AP.2.1a ***Students will use pseudocode and/or flowcharts to organize and sequence an algorithm that addresses a complex problem, even though they may not actually program the solutions.*** *For example, students might express an algorithm that produces a recommendation for purchasing sneakers based on inputs such as size, colors, brand, comfort, and cost. Testing the algorithm with a wide range of inputs and users allows students to refine their recommendation algorithm and to identify other inputs they may have initially excluded.*

AP.2.2 **Create clearly named variables that represent different data types and perform operations on their values. [VARIABLES] (P5.1, P5.2)**

A variable is like a container with a name, in which the contents may change, but the name (identifier) does not.

AP.2.2a ***When planning and developing programs, students should decide when and how to declare and name new variables.*** *Examples of operations include adding*

points to the score, combining user input with words to make a sentence, changing the size of a picture, or adding a name to a list of people.

AP.2.2b ***Students should use naming conventions to improve program readability.***

AP.2.3 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals. [CONTROL] (P5.1, P5.2)

Control structures can be combined in many ways. Nested loops are loops placed within loops. Compound conditionals combine two or more conditions in a logical relationship (e.g., using AND, OR, and NOT), and nesting conditionals within one another allows the result of one conditional to lead to another.

AP.2.3a ***Students will design and develop programs that combine control structures.*** *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.*

AP.2.4 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. [MODULARITY] (P3.2)

Decomposition facilitates aspects of program development by allowing students to focus on one piece at a time (e.g., getting input from the user, processing the data, and displaying the result to the user). Decomposition also enables different students to work on different parts at the same time.

AP.2.4a ***Students should break down problems into subproblems, which can be further broken down to smaller parts.*** *For example, animations can be decomposed into multiple scenes, which can be developed independently.*

AP.2.5 Create procedures with parameters to organize code and make it easier to reuse. [MODULARITY] (P4.1, P4.3)

AP.2.5a ***Students will create procedures and/or functions that are used multiple times within a program to repeat groups of instructions.*** *These procedures can be generalized by defining parameters that create different outputs for a wide range of inputs. For example, a procedure to draw a circle involves many instructions, but all of them can be invoked with one instruction, such as "drawCircle." By adding a radius parameter, the user can easily draw circles of different sizes.*

AP.2.6 Seek and incorporate feedback from team members and users to refine a solution that meets user needs. [PROGRAM DEVELOPMENT] (P2.3, P1.1)

Development teams that employ user-centered design create solutions (e.g., programs and devices) that can have a large societal impact, such as an app that allows people with speech difficulties to translate hard-to-understand pronunciation into understandable language.

AP.2.6a ***Students should begin to seek diverse perspectives throughout the design process to improve their computational artifacts.*** *Considerations of the end user may include usability, accessibility, age-appropriate content, respectful language, user perspective, pronoun use, color contrast, and ease of use.*

AP.2.7 Incorporate existing code, media, and libraries into original programs and give attribution. [PROGRAM DEVELOPMENT] (P4.2, P5.2, P7.3)

Building on the work of others enables students to produce more interesting and powerful creations.

AP.2.7a ***Students should use portions of code, algorithms, and/or digital media in their own programs and websites.*** *At this level, they may also import libraries and connect to web application program interfaces (APIs). For example, when creating a side-scrolling games, students may incorporate portions of code that create a realistic jump movement from another person's game, and they may*

also import Creative Commons-lessened images to use in the background.

AP.2.7b ***Students should give attribution to the original creator's contributions.***

AP.2.8 Systematically test and refine programs using a range of test cases. [PROGRAM DEVELOPMENT] (P6.1)

Test cases are created and analyzed to better meet the needs of users and to evaluate whether programs function as intended. At this level, testing should become a deliberate process that is more iterative, systematic, and proactive than at lower levels.

AP.2.8a ***Students will test programs by considering potential errors, such as what will happen if a user enters invalid input (e.g., negative numbers and zero instead of positive numbers).***

AP.2.9 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts. [PROGRAM DEVELOPMENT] (P2.2)

Collaboration is a common and crucial practice in programming development. Often, many individuals and groups work on the interdependent parts of a project together.

AP.2.9a ***Students will work collaboratively in groups.***

AP.2.9b ***Students should assume predefined roles within their teams and manage the project workflow using structured timelines.*** *With teacher guidance, they will begin to create collective goals, expectations, and equitable workloads. For example, students may divide the design stage of a game into planning the storyboard, flowchart, and different parts of the game mechanics. They can then distribute tasks and roles among members of the team and assign deadlines.*

AP.2.9c ***Students should give attribution to the original creators to acknowledge their contributions.***

AP.2.10 Document programs in order to make them easier to follow, test, and debug. [PROGRAM DEVELOPMENT] (P7.2)

Documentation allows creators and others to more easily use and understand a program.

AP.2.10a ***Students should provide documentation for end users that explains their artifacts and how they function.*** *For example, students could provide a project overview and clear user instructions.*

AP.2.10b ***Students should incorporate comments in their product (comments in the code).***

AP.2.10c ***Students should communicate their process using design documents, flowcharts, and presentations.***

Level 2: GRADES 6-8 - Impacts of Computing

Impacts of Computing (IC.2)

Conceptual understanding: Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and responsible person should understand the social implications of the digital world, including equity and access to computing.

IC.2.1 Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options. [CULTURE] (P7.2)

Advancements in computer technology are neither wholly positive nor negative; however, the ways that people use computing technologies have tradeoffs.

IC.2.1a ***Students should consider current events related to broad ideas, including privacy, communication, and automation.*** *For example, driverless cars can*

increase convenience and reduce accidents, but they are also susceptible to hacking. The emerging industry will not only reduce the number of taxi and shared-ride drivers but also create more software engineering and cybersecurity jobs.

IC.2.2 Discuss issues of bias and accessibility in the design of existing technologies. [CULTURE] (P1.2)

IC.2.2a ***Students should test and discuss the usability of various technology tools (e.g., apps, games, and devices) with the teacher's guidance.*** *For example, facial recognition software that works better for lighter skin tones was likely developed with a homogeneous testing group and could be improved by sampling a more diverse population. When discussing accessibility, students may notice that allowing a user to change font sizes and colors will not only make an interface usable for people with low vision but also benefits users in various situations, such as in bright daylight or a dark room.*

IC.2.3 Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact. [SOCIAL INTERACTIONS] (P2.4, P5.2)

Crowdsourcing is gathering services, ideas, or content from a large group of people, especially from the online community. It can be done at the local level (e.g., classroom or school) or global level (e.g., age-appropriate online communities, like Scratch and Minecraft).

IC.2.3a ***Students should collaborate with many contributors.*** *For example, a group of students could combine animations to create a digital community mosaic. They could also solicit feedback from many people though use of online communities and electronic surveys.*

IC.2.4 Describe tradeoffs between allowing information to be public and keeping information private and secure. [SAFETY, LAW, & ETHICS] (P7.2)

Sharing information online can help establish, maintain, and strengthen connections between people. For example, it allows artists and designers to display their talents and reach a broad audience; however, security attacks often start with personal information that is publicly available online. Social engineering is based on tricking people into revealing sensitive information and can be thwarted by being wary of attacks, such as phishing and spoofing.

IC.2.4a ***Students should discuss and describe the benefits and dangers of allowing information to be public or kept private and secure.***

Level 3A: GRADES 9-10

Level 3A: GRADES 9-10 - Computing Systems

Computing Systems (CS.3A)

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

CS.3A.1 Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects. [DEVICES] (P4.1)

Computing devices are often integrated with other systems, including biological, mechanical, and social systems. A medical device can be embedded inside a person to monitor and regulate his or her health, a hearing aid (a type of assistive device) can filter out certain frequencies and magnify others, a monitoring device installed in a motor vehicle can track a person's driving patterns and habits, and a facial recognition device can be integrated into a security system to identify a person. The creation of integrated or embedded systems is not an expectation at this level.

CS.3A.1a ***Students should be able to identify embedded computer systems.***

CS.31.1b ***Students should describe the types of data and procedures that are included in the embedded system and explain how the implementation details are hidden from the user.*** *For example, a students might select a car stereo, identify the types of data (radio station presets, station name or number, volume level) and procedures (increase volume, store/recall saved station, mute) it includes.*

CS.3A.2 Compare levels of abstraction and interactions between application software, system software, and hardware layers. [HARDWARE & SOFTWARE] (P4.1)

At its most basic level, a computer is composed of physical hardware and electrical impulses. Multiple layers of software are built upon the hardware and interact with the layers above and below them to reduce complexity. System software manages a computing device's resources so that software can interact with hardware. System software is used on many different types of devices, such as smart TVs, assistive devices, virtual components, cloud components, and drones. For example, students may explore the progression from voltage to binary signal to logic gates to adders and so on. Knowledge of specific, advanced terms for computer architecture, such as BIOS, kernel, or bus, is not expected at this level.

CS.3A.2a ***Students should be able to distinguish between hardware and software.***

CS.3A.2b ***Students should be able to describe the purpose of and differences between system software (i.e. operating system) and application software (i.e. word processor).***

CS.3A.2c ***Students should be able to describe how software and hardware interact.*** *For example, text-editing software interacts with the operating system to receive input from the keyboard, convert the input to bits for storage, and interpret the bits as readable text to display on the monitor.*

CS.3A.3 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors. [TROUBLESHOOTING] (P6.2)

Troubleshooting complex problems involves the use of multiple sources when researching, evaluating, and implementing potential solutions. Troubleshooting also relies on experience,

such as when people recognize that a problem is similar to one they have seen before or adapt solutions that have worked in the past. Examples of complex troubleshooting strategies include resolving connectivity problems, adjusting system configurations and settings, ensuring hardware and software compatibility, and transferring data from one device to another.

CS.3A.3a ***Students should develop guidelines by creating an artifact that conveys systematic troubleshooting strategies (i.e. create a flow chart or a job aid for a help desk employee).***

Level 3A: GRADES 9-10 - Networks and the Internet

Networks and the Internet (NI.3A)

Conceptual understanding: Computing devices typically do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing. Networks and communication systems provide greater connectivity in the computing world by providing fast, secure communication and facilitating innovation.

NI.3A.1 Evaluate the scalability and reliability of networks by describing the relationship between routers, switches, servers, topology, and addressing. [NETWORK COMMUNICATION & ORGANIZATION] (P4.1)

Each device is assigned an address that uniquely identifies it on the network. Routers function by comparing IP addresses to determine the pathways packets should take to reach their destination. Switches function by comparing MAC addresses to determine which computers or network segments will receive frames. Students could use online network simulators to experiment with these factors.

NI.3A.1a ***Students should be able to define a MAC address – what is it and how it is used.***

NI.3A.1b ***Students should be able to explain what a router and a switch are and how they work inside a network.***

NI.3A.1c ***Students should be able to define what a server is and how it is used in a network.***

NI.3A.1d ***Students should be able to list various types of network topology and explain why each is used.***

NI.3A.1e ***Students should be able to verbally and visually explain how addressing, routers, switches, and servers all work together in a network.***

NI.3A.2 Give examples to illustrate how sensitive data can be affected by malware and other attacks. [CYBERSECURITY] (P7.2)

Network security depends on a combination of hardware, software, and practices that control access to data and systems. The needs of users and the sensitivity of data determine the level of security implemented. Potential security problems, such as denial-of-service attacks, ransomware, viruses, worms, spyware, and phishing, present threats to sensitive data.

NI.3A.2a ***Students should be able to discuss how sensitive data can be affected by malware and other attacks.*** *Students might reflect on case studies or current events in which governments or organizations experienced data leaks or data loss as a result of these types of attacks.*

NI.3A.3 Recommend security measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts. [CYBERSECURITY] (P3.1, 3.3)

Security measures may include physical security tokens, two-factor authentication, and biometric verification. Potential security problems, such as denial-of-service attacks, ransomware, viruses, worms, spyware, and phishing, exemplify why sensitive data should be

securely stored and transmitted. The timely and reliable access to data and information services by authorized users, referred to as availability, is ensured through adequate bandwidth, backups, and other measures.

NI.3A.3a ***Students should understand the different types of security problems and the different types of devices that can be impacted.*** *Potential security problems may include issues such as denial-of-service attacks, ransomware, viruses, worms, spyware, phishing, and social engineering. Some types of devices impacted may include laptops, tablets, cell phones, self-driving cars, ATMs, and others.*

NI.3A.3b ***Students should systematically evaluate different security measures based on efficiency, feasibility, and ethical impacts.*** *Students might address issues such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns.*

NI.3A.4 Compare various security measures considering tradeoffs between the usability and security of a computing system. [CYBERSECURITY] (P6.3)

Security measures may include physical security tokens, two-factor authentication, and biometric verification, but choosing security measures involves tradeoffs between the usability and security of the system. The needs of users and the sensitivity of data determine the level of security implemented.

NI.3A.4a ***Students should be able to explain different types of security measures and discuss the tradeoffs between usability and security.*** *For example, students might discuss computer security policies in place at the local level that present a tradeoff between usability and security, such as a web filter that prevents access to many educational sites but keeps the campus network safe.*

NI.3A.5 Explain tradeoffs when selecting and implementing cybersecurity recommendations. [CYBERSECURITY] (P7.2)

Network security depends on a combination of hardware, software, and practices that control access to data and systems. The needs of users and the sensitivity of data determine the level of security implemented. Every security measure involves tradeoffs between the accessibility and security of the system.

NI.3A.5a ***Students should be able to describe, justify, and document choices they make using terminology appropriate for the intended audience and purpose.*** *Students could debate issues from the perspective of diverse audiences, including individuals, corporations, privacy advocates, security experts, and government.*

Level 3A: GRADES 9-10 - Data and Analysis

Data and Analysis (DA.3A)

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

DA.3A.1 Translate between different bit representations of real-world phenomena, such as characters, numbers, and images. [STORAGE] (P4.1)

DA.3A.1a ***Students should be able to translate between different bit representations.*** *For example, convert hexadecimal color codes to decimal percentages, ASCII/Unicode representation, or converting binary to base 10.*

DA.3A.1b ***Students should be able to discuss how data sequences can be interpreted in a variety of formats.*** *For example, text, numbers, sound, and images.*

DA.3A.2 Evaluate the tradeoffs in how data elements are organized and where data is stored. [STORAGE] (P3.3)

People make choices about how data elements are organized and where data is stored. These choices affect cost, speed, reliability, accessibility, privacy, and integrity.

DA.3A.2a ***Students should evaluate whether a chosen solution is most appropriate for a particular problem.*** *Students might consider the cost, speed, reliability, accessibility, privacy, and integrity tradeoffs between storing photo data on a mobile device versus in the cloud.*

DA.3A.3 Collect, transform, and organize data to help others better understand a problem. [COLLECTION, VISUALIZATION, & TRANSFORMATION] (P4.4)

People transform, generalize, simplify, and present large data sets in different ways to influence how other people interpret and understand the underlying information. Examples include visualization, aggregation, rearrangement, and application of mathematical operations. People use software tools or programming to create powerful, interactive data visualizations and perform a range of mathematical operations to transform and analyze data.

DA.3A.3a ***Students should use various data collection techniques for different types of computational problems.*** *For example, user surveys, mobile device GPS, social media data sets, etc.*

DA.3A.3b ***Use computational tools to collect, transform, and organize data to help others better understand a problem.***

DA.3A.3c ***Students should use data analysis to identify significant patterns in data sets.***

DA.3A.4 Create and evaluate computational models that represent real-world systems. [INFERENCE & MODELS] (P4.4)

Computational models make predictions about processes or phenomenon based on selected data and features. The amount, quality, and diversity of data and the features chosen can affect the quality of a model and ability to understand a system. Predictions or inferences are tested to validate models.

DA.3A.4a ***Students should create computational models that simulate real-world systems (e.g., ecosystems, epidemics, spread of disease).***

DA.3A.4b ***Students should analyze and evaluate the ability of models and simulations to formulate, refine, and test hypothesis.***

Level 3A: GRADES 9-10 - Algorithms and Programming

Algorithms and Programming (AP.3A)

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

AP.3A.1 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests. [ALGORITHMS] (P5.2)

A prototype is a computational artifact that demonstrates the core functionality of a product or process. Prototypes are useful for getting early feedback in the design process and can yield insight into the feasibility of a product. The process of developing computational artifacts

embraces both creative expression and the exploration of ideas to create prototypes and solve computational problems.

AP.3A.1a ***Students create artifacts that are personally relevant or beneficial to their community and beyond.*** *Students should develop artifacts in response to a task or a computational problem that demonstrate the performance, reusability, and ease of implementation of an algorithm.*

AP.3A.2 Use lists and functions to simplify solutions, generalizing computational problems instead of repeatedly using simple variables. [VARIABLES] (P4.1)

AP.3A.2a ***Students should be able to identify common features in multiple segments of code and substitute a single segment that uses lists (arrays) or functions to account for the differences.***

AP.3A.3 Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made. [CONTROL] (P5.2)

Implementation includes the choice of programming language, which affects the time and effort required to create a program. Readability refers to how clear the program is to other programmers and can be improved through documentation. The discussion of performance is limited to a theoretical understanding of execution time and storage requirements; a quantitative analysis is not expected. Control structures at this level may include conditional statements, loops, event handlers, and recursion.

AP.3A.3a ***Students should be able to justify by explaining the benefits and drawbacks of the selection of specific control structures with regard to implementation, readability, and program performance.*** *For example, students might compare the readability and program performance of iterative and recursive implementations of procedures that calculate the Fibonacci sequence.*

AP.3A.4 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions. [CONTROL] (P5.2)

In this context, relevant computational artifacts include programs, mobile apps, or web apps. Events can be user-initiated, such as a button press, or system-initiated, such as a timer firing. At previous levels, students have learned to create and call procedures. Here, students design procedures that are called by events.

AP.3A.4a ***Students will design procedures that are called by events.*** *Students might create a mobile app that updates a list of nearby points of interest when the device detects that its location has been changed.*

AP.3A.5 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects. [MODULARITY] (P3.2)

AP.3A.5a ***Students should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist.*** *For example, students could create an app to solve a community problem by connecting to an online database through an application programming interface (API).*

AP.3A.6 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs. [MODULARITY] (P5.2)

Computational artifacts can be created by combining and modifying existing artifacts or by developing new artifacts. Examples of computational artifacts include programs, simulations, visualizations, digital animations, robotic systems, and apps. Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall

purpose. Modules allow for better management of complex tasks. The focus at this level is understanding a program as a system with relationships between modules.

AP.3A.6a ***Students will create artifacts by using procedures within a program, combinations of data, and procedures, or independent but interrelated programs.*** *The choice of implementation, such as programming language or paradigm, may vary. Students could incorporate computer vision libraries to increase the capabilities of a robot or leverage open-source JavaScript libraries to expand the functionality of a web application.*

AP.3A.7 Systematically design and develop programs for broad audiences by incorporating feedback from users. [PROGRAM DEVELOPMENT] (P5.1)

Examples of programs could include games, utilities, and mobile applications. Students at lower levels collect feedback and revise programs.

AP.3A.1b ***Students should do so through a systematic process that includes feedback from broad audiences.*** *Students might create a user satisfaction survey and brainstorm distribution methods that could yield feedback from a diverse audience, documenting the process they took to incorporate selected feedback in product revisions.*

AP.3A.8 Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries. [PROGRAM DEVELOPMENT] (P7.3)

Examples of software licenses include copyright, freeware, and the many open-source licensing schemes. At previous levels, students adhered to licensing schemes.

AP.3A.8a ***Students should consider licensing implications for their own work, especially when incorporating libraries and other resources.*** *Students might consider two software libraries that address a similar need, justifying their choice based on the library that has the least restrictive license.*

AP.3A.9 Evaluate and refine computational artifacts to make them more usable and accessible. [PROGRAM DEVELOPMENT] (P6.3)

Testing and refinement is the deliberate and iterative process of improving a computational artifact. This process includes debugging (identifying and fixing errors) and comparing actual outcomes to intended outcomes.

AP.3A.9a ***Students should respond to the changing needs and expectations of end users and improve the performance, reliability, usability, and accessibility of artifacts.*** *For example, students could incorporate feedback from a variety of end users to help guide the size and placement of menus and buttons in a user interface.*

AP.3A.10 Design and develop computational artifacts working in team roles using collaborative tools. [PROGRAM DEVELOPMENT] (P2.4)

Collaborative tools could be as complex as source code version control system or as simple as a collaborative word processor. Team roles in pair programming are driver and navigator but could be more specialized in larger teams. As programs grow more complex, the choice of resources that aid program development becomes increasingly important and should be made by the students.

AP.3A.10a ***Students will work in teams using collaborative tools to design and develop computational artifacts.*** *Students might work as a team to develop a mobile application that addresses a problem relevant to the school or community, selecting appropriate tools to establish and manage the project timeline; design, share, and revise graphical user interface elements; and track planned, in-progress, and completed components.*

AP.3A.11 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs. [PROGRAM DEVELOPMENT] (P7.2)

Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program; combinations of data and procedures; or independent, but interrelated, programs. The development of complex programs is aided by resources such as libraries and tools to edit and manage parts of the program.

AP.3A.11a ***Students will document design decisions using text, graphics, presentations, and/or demonstrations.***

Level 3A: GRADES 9-10 - Impacts of Computing

Impacts of Computing (IC.3A)

Conceptual understanding: Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and responsible person should understand the social implications of the digital world, including equity and access to computing.

IC.3A.1 Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices. [CULTURE] (P1.2)

Computing may improve, harm, or maintain practices. Equity deficits, such as minimal exposure to computing, access to education, and training opportunities, are related to larger, systemic problems in society.

IC.3A.1a ***Students should be able to evaluate the accessibility of a product to a broad group of end users, such as people who lack access to broadband or who have various disabilities.***

IC.3A.2b ***Students should also begin to identify potential bias during the design process to maximize accessibility in product design.***

IC.3A.2 Test and refine computational artifacts to reduce bias and equity deficits. [CULTURE] (P1.2)

Biases could include incorrect assumptions developers have made about their user base. Equity deficits include minimal exposure to computing, access to education, and training opportunities.

IC.3A.2a ***Students should begin to identify potential bias during the design process to maximize accessibility in product design and become aware of professionally accepted accessibility standards to evaluate computational artifacts for accessibility.***

IC.3A.3 Demonstrate ways a given algorithm applies to problems across disciplines. [CULTURE] (P3.1)

Computation can share features with disciplines, such as art and music, by algorithmically translating human intention into an artifact.

IC.3A.3a ***Students should be able to identify real-world problems that span multiple disciplines, such as increasing bike safety with new helmet technology, and that can be solved computationally.***

IC.3A.4 Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields. [SOCIAL INTERACTIONS] (P2.4)

Many aspects of society, especially careers, have been affected by the degree of communication afforded by computing. The increased connectivity between people in different cultures and in different career fields has changed the nature and content of many careers.

IC.3A.4a ***Students should explore different collaborative tools and methods used to solicit input from team members, classmates, and others, such as participation in online forums or local communities.*** *For example, students could compare ways different social media tools could help a team become more cohesive*

IC.3A.5 Explain the beneficial and harmful effects that intellectual property laws can have on innovation. [SAFETY, LAW, & ETHICS] (P7.3)

Laws govern many aspects of computing, such as privacy, data, property, information, and identity. These laws can have beneficial and harmful effects, such as expediting or delaying advancements in computing and protecting or infringing upon people's rights. International differences in laws and ethics have implications for computing. For examples, laws that mandate the blocking of some file-sharing websites may reduce online piracy but can restrict the right to access information. Firewalls can be used to block harmful viruses and malware but can also be used for media censorship.

IC.3A.5a ***Students should be aware of intellectual property laws and be able to explain how they are used to protect the interests of innovators and how patent trolls abuse the laws for financial gain.***

IC.3A.6 Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users. [SAFETY, LAW, & ETHICS] (P7.2)

Data can be collected and aggregated across millions of people, even when they are not actively engaging with or physically near the data collection devices. This automated and nonevident collection can raise privacy concerns, such as social media sites mining an account even when the user is not online. Other examples include surveillance video used in a store to track customers for security or information about purchase habits or the monitoring of road traffic to change signals in real time to improve road efficiency without drivers being aware. Methods and devices for collecting data can differ by the amount of storage required, level of detail collected, and sampling rates.

IC.3A.6a ***Students should be able to explain the privacy concerns related to the collection and generation of data through automated processes.***

IC.3A.7 Evaluate the social and economic implications of privacy in the context of safety, law, or ethics. [SAFETY, LAW, & ETHICS] (P7.3)

Laws govern many aspects of computing, such as privacy, data, property, information, and identity. International differences in laws and ethics have implications for computing.

IC.3A.7a ***Students should evaluate the social and economic implications of privacy in the context of safety, law, or ethics.*** *For example, students might review case studies or current events that present an ethical dilemma when an individual's right to privacy is at odds with the safety, security, or wellbeing of a community.*

Level 3B: GRADES 11-12

Level 3B: GRADES 11-12 - Computing Systems

Computing Systems (CS.3B)

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

CS.3B.1 **Categorize the roles of operating system software.**
Examples of roles could include memory management, data storage/retrieval, processes management, and access control. **[HARDWARE & SOFTWARE]** **(P7.2)**

CS.3B.2 **Illustrate ways computing systems implement logic, input, and output through hardware components.**
Examples of components could include logic gates and IO pins. **[TROUBLESHOOTING]** **(P7.2)**

Level 3B: GRADES 11-12 - Networks and the Internet

Networks and the Internet (NI.3B)

Conceptual understanding: Computing devices typically do not operate in isolation. Networks connect computing devices to share information and resources and are an increasingly integral part of computing. Networks and communication systems provide greater connectivity in the computing world by providing fast, secure communication and facilitating innovation.

NI.3B.1 **Describe the issues that impact network functionality (e.g., bandwidth, load, delay, topology).**
Recommend use of free online network simulators to explore how these issues impact network functionality. **[NETWORK COMMUNICATION & ORGANIZATION]** **(P7.2)**

NI.3B.2 **Compare ways software developers protect devices and information from unauthorized access.**
Examples of security concerns to consider: encryption and authentication strategies, secure coding, and safeguarding keys. **[CYBERSECURITY]** **(P7.2)**

Level 3B: GRADES 11-12 - Data and Analysis

Data and Analysis (DA.3B)

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

DA.3B.1 **Use data analysis tools and techniques to identify patterns in data representing complex systems.**
For example, identify trends in a dataset representing social media interactions, movie reviews, or shopping patterns. **[COLLECTION, VISUALIZATION, & TRANSFORMATION] (P4.1)**

DA.3B.2 **Select data collection tools and techniques to generate data sets that support a claim or communicate information. [COLLECTION, VISUALIZATION, & TRANSFORMATION] (P7.2)**

DA.3B.3 **Evaluate the ability of models and simulations to test and support the refinement of hypotheses. [INFERENCE & MODELS] (P4.4)**

Level 3B: GRADES 11-12 - Algorithms and Programming

Algorithms and Programming (AP.3B)

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

AP.3B.1 **Describe how artificial intelligence drives many software and physical systems.**
Examples include digital ad delivery, self-driving cars, and credit card fraud detection. **[ALGORITHMS] (P7.2)**

AP.3B.2 **Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.**
Games do not have to be complex. Simple guessing games, Tic-Tac-Toe, or simple robot commands will be sufficient. **[ALGORITHMS] (P5.3)**

AP.3B.3 **Use and adapt classic algorithms to solve computational problems.**
Examples could include sorting and searching. **[ALGORITHMS] (P4.2)**

AP.3B.4 **Evaluate algorithms in terms of their efficiency, correctness, and clarity.**
Examples could include sorting and searching. **[ALGORITHMS] (P4.2)**

AP.3B.5 **Compare and contrast fundamental data structures and their uses.**
Examples could include strings, lists, arrays, stacks, and queues. **[VARIABLES] (P4.2)**

AP.3B.6 **Illustrate the flow of execution of a recursive algorithm. [CONTROL] (P3.2)**

AP.3B.7 **Construct solutions to problems using student-created components, such as procedures, modules and/or objects.**
Object-oriented programming is optional at this level. Problems can be assigned or student-selected. **[MODULARITY] (P5.2)**

AP.3B.8 **Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution.**
As students encounter complex, real-world problems that span multiple disciplines or social systems, they should decompose complex problems into manageable sub problems that could potentially be solved with programs or procedures that already exist. For example, students could create an app to solve a community problem by connecting to an online database through an application programming interface (API). **[MODULARITY] (P4.1)**

AP.3B.9 **Demonstrate code reuse by creating programming solutions using libraries and APIs.**
Libraries and APIs can be student-created or common graphics libraries or maps APIs, for example. **[MODULARITY] (P5.3)**

AP.3B.10 **Plan and develop programs for broad audiences using a software life cycle process.**
Processes could include agile, spiral, or waterfall. **[PROGRAM DEVELOPMENT] (P5.1)**

AP.3B.11 **Explain security issues that might lead to compromised computer programs.**
For example, common issues include lack of bounds checking, poor input validation, and circular references. **[PROGRAM DEVELOPMENT] (P7.2)**

AP.3B.12 **Develop programs for multiple computing platforms.**
Example platforms could include: computer desktop, web, or mobile. **[PROGRAM DEVELOPMENT] (P5.2)**

AP.3B.13 **Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (code documentation) in a group software project.**
Group software projects can be assigned or student-selected. **[PROGRAM DEVELOPMENT] (P2.4)**

AP.3B.14 **Develop and use a series of test cases to verify that a program performs according to its design specifications.**
At this level, students are expected to select their own test cases. **[PROGRAM DEVELOPMENT] (P6.1)**

AP.3B.15 **Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).**
For instance, changes made to a method or function signature could break invocations of that method elsewhere in a system. **[PROGRAM DEVELOPMENT] (P5.3)**

AP.3B.16 **Evaluate key qualities of a program through a process such as a code review.**
Examples of qualities could include correctness, usability, readability, efficiency, portability and scalability. **[PROGRAM DEVELOPMENT] (P6.3)**

AP.3B.17 **Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems.**
Examples of features include blocks versus text, indentation versus curly braces, and high-level versus low-level. **[PROGRAM DEVELOPMENT] (P7.2)**

Level 3B: GRADES 11-12 - Impacts of Computing

Impacts of Computing (IC.3B)

Conceptual understanding: Computing affects many aspects of the world in both positive and negative ways at local, national, and global levels. Individuals and communities influence computing through their behaviors and cultural and social interactions, and in turn, computing influences new cultural practices. An informed and responsible person should understand the social implications of the digital world, including equity and access to computing.

IC.3B.1 **Evaluate computational artifacts to maximize their beneficial effects and minimize harmful effects on society. [CULTURE] (P6.1, 1.2)**

IC.3B.2 **Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society. [CULTURE] (P1.2)**

IC.3B.3 **Predict how computational innovations that have revolutionized aspects of our culture might evolve.**
Areas to consider might include education, healthcare, art/entertainment, and energy.
[CULTURE] (P7.2)

IC.3B.4 **Debate laws and regulations that impact the development and use of software.**
Areas to consider might include education, healthcare, art/entertainment, energy, and artificial intelligence (for example ethical concerns around self-driving cars). **[SAFETY, LAW, & ETHICS] (P7.3, 3.3)**

Support Documents and Resources

The MDE will develop support documents after these standards have been approved by the State Board of Education. Local districts, schools, and teachers may use these documents to construct standards-based computer science curriculum, allowing them to customize content to fit their students' needs and match available instructional materials. The support documents will include suggested resources, instructional strategies, and essential knowledge.

Professional development efforts will be aligned with the standards and delivered in accord with teacher resources to help expand expertise in delivering student-centered lessons (e.g., inquiry-based learning, 5-E instructional models, or other best practices in STEM teaching). The most successful national models and programs will be referenced for a capacity-building effort that can develop a more effective culture of computer science education in Mississippi.

References

Aho, A.V. (2011, January) Computation and Computational Thinking. *ACM Ubiquity, 1*, 1-8.

Brennan, K., & Resnick, M. (2012). *Using artifact-based interviews to study the development of computational thinking in interactive media design*. Paper presented at the annual meeting of the American Educational Research Association, Vancouver, BC, Canada.

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing*, *1*, 67–69.

Computer Science Teachers Association (2017). *CSTA K-12 Computer Science Standards, Revised 2017.* Retrieved from http://www.csteachers.org/standards.

Cuny, J., Snyder, L., & Wing, J.M. (2010). *Demystifying computational thinking for noncomputer scientists.* Unpublished manuscript in progress. Retrieved from http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Google/Gallup. (2015). *Searching for Computer Science: Access and Barriers in U.S. K-12 Education*. Retrieved from https://services.google.com/fh/files/misc/searching-for-computer-science_report.pdf

K-12 Computer Science Framework. (2016). Retrieved from http://www.k12cs.org.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. NY: Basic Books.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society, 366(1881)*, 3717–3725.

Tilley-Coulson, Eve (2016). *Policy Update National Association of State Boards of Education: State Move toward Computer Science Standards*, (2016, June). Vol. 23, No. 17.