

APPENDIX D

Teacher Resource Progression Document to the 2019 Wyoming Computer Science Standards



Appendix D is provided as an optional resource to support teachers as they implement these new standards. This document contains Grade Level Benchmark Progressions and Cross-Disciplinary Connections that directly tie to other Wyoming Content Standards.

Companion Document to the 2019 CS Standards

The [2019 Computer Science Standards](https://edu.wyoming.gov/standards/computer-science) document is found at edu.wyoming.gov/standards/computer-science

TO BE FULLY IMPLEMENTED IN DISTRICTS BY THE BEGINNING OF SCHOOL YEAR 2022-23

ACKNOWLEDGEMENT

The Wyoming State Board of Education would like to thank the Wyoming Department of Education, as well as educators, parents and community members, business and industry representatives, community college representatives, and the University of Wyoming representatives for their help with the development of these computer science standards.

**Jillian Balow, Superintendent of Public Instruction
Wyoming Department of Education**

**Laurie Hernandez, Division Director
Standards and Assessment Division**

**Barb Marquer, Standards Team Supervisor
Brian Cole and Catherine Palmer —WDE Facilitators**

**Wyoming Department of Education
122 E. 25th Street, Suite E200
Cheyenne, WY 82002-0050**



The Wyoming Department of Education does not discriminate on the basis of race, color, national origin, sex, age, or disability in admission or access to, or treatment or employment in its educational programs or activities. Inquiries concerning Title VI, Title IX, Section 504, and the Americans with Disabilities Act may be referred to the Wyoming Department of Education, Office for Civil Rights Coordinator, 2nd floor, Hathaway Building, Cheyenne, Wyoming 82002-0050 or (307) 777-6252, or the Office for Civil Rights, Region VIII, U. S. Department of Education, Federal Building, Suite 310, 1244 Speer Boulevard, Denver,

Computer Science Standards Review Committee



Andrea Burrows—University of Wyoming

Barbara Sanchez—Teton CSD #1

Carla Hester Croff—Western Wyoming Community College

Christopher Larsen—UW Student (CS major)

Connie Poulsen-Hollin—Platte CSD #2

Denise Miller—Natrona CSD #1

Dennis Fischer—Platte CSD #1

E. Anne Gunn—NWCCD / Sheridan College

Elizabeth David—Sublette CSD #1

Emily Vercoe—UW, Wyoming EPSCoR

Erin Moore—Gannett Peak Technical Services

Ernest (Wade) Williams—Business Member, Jackson

Greg Bianchi—Microsoft, Inc.

Harmony Davidson—Carbon CSD #2

Janel Korhonen-Goff—Parent, Casper

Jared O’Leary—BootUp PD

Jason Wheeler—Park CSD #16

Jenifer Albrandt—Converse CSD #2

Joanne Flanagan—Fremont CSD #25

Joel E. Hansen—Cognizant Technology Solutions

John Drecher—Apple Inc.

Joshua C. Sanderlin—Array School of Technology & Design

Joshua Michelson—First Interstate Bank—Jackson

Karen Rogers—Wyoming Game and Fish Dept.

Katie Alvarez—Big Horn CSD #1

Krista Sweckard—Johnson CSD #1

KyLee Shoemaker—Weston CSD #1

Lindsey Allbright—Niobrara CSD #1

London Jenks—Hot Springs CSD #1

Lucy Martin—Torrington

Nancy Nelson—Big Horn CSD #3

Becky Byer—Natrona CSD #1

Rodney Brown—Wyre Enterprises, LLC—Consulting Services

Ruben Gamboa—University of Wyoming

Sandra Ahlstrom—Sheridan CSD #2

Sara Schnell—Laramie CSD #2

Sean Roberts—Code.org

Sonya Tysdal—Weston CSD #1

Stephen Nelson—Sublette CSD #9

Zachary Opps—Park CSD #1

INTRODUCTION:

The Wyoming Computer Science Content and Performance Standards (WYCPS) were developed in accordance with Wyoming State Statute W.S. 21-2-304(c). The 2019 Wyoming Computer Science Standards were developed collaboratively through the contributions of the Computer Science Standards Review Committee (CSSRC) which included Wyoming parents, educators, and community members, as well as business members from across the state and nation. The committee's work was informed and guided by initial public input through community forums, as well as input solicited from specific stakeholder groups.

RATIONALE:

The committee's (CSSRC) vision is that every student in every school has the opportunity to learn computer science. We believe that computing is fundamental to understanding and participating in an increasingly technological society, and it is essential for every Wyoming student to learn as part of a modern education. We see computer science as a subject that provides students with a critical lens for interpreting the world around them and challenges them to explore how computing and technology can expand Wyoming's impact on the world.

The standards we (CSSRC) present here provide the necessary foundation for local school district decisions about curriculum, assessment, and instruction. Implementation of these standards will better prepare Wyoming high school graduates for the rigors of college and/or career. In turn, Wyoming employers will be able to hire workers with a strong foundation in Computer Science—both in specific content areas and in critical thinking and inquiry-based problem solving.

In grades K-8, the committee (CSSRC) provides suggested progressions embedded within each grade band. The purpose is to show how each grade level could address the standard in a sequential and logical manner as well as to emphasize the importance of repetition of specific skills. Assessments

should align to the end-of-grade-band benchmark, highlighted in gold on the right-hand side of the document.

In grades 9-12, the committee provides level 1 and level 2 benchmarks. Level 1 benchmarks include introductory skills. The level 2 benchmarks are intended for students who wish to advance their study of Computer Science. All level 1 and level 2 benchmarks are intended to be assessed for students taking courses covering the skills described in the benchmark.

ORGANIZATION OF THE COMPUTER SCIENCE (CS) STANDARDS:

Content Standards

Content standards define what students are expected to know and be able to do throughout their study of computer science. They do not dictate what methodology or instructional materials should be used, nor how the material is delivered.

Benchmarks

Benchmarks are the skills students must master in order to demonstrate proficiency of the content standards throughout the grade band. In this standards document, you will find end-of-grade band benchmarks for grades K-8, along with suggested progressions for meeting the end-of-grade band benchmark, highlighted in gold. In grades 9-12, benchmarks are organized into 2 levels. Mostly, Level 1 is intended to represent the introductory level while Level 2 reaches a deeper level.

Performance Level Descriptors (PLDs)

Performance Level Descriptors (PLDs) describe the performance expectations of students for each of the four (4) performance level categories: advanced, proficient, basic, and below basic.

Clarification Statement

Statements which provide further explanation or examples to support teachers in instruction.

Domain

The core concepts to be studied in computer science are as follows: 1) Computing Systems; 2) Networks and the Internet; 3) Data and Analysis; 4) Algorithms and Programming; and 5) Impacts of Computing.

WYOMING 2019 COMPUTER SCIENCE DOMAINS & STANDARDS

Computing Systems	Networks & The Internet	Data Analysis	Algorithms & Programming	Impacts of Computing
CS.D—Devices CS.HS—Hardware & Software CS.T—Troubleshooting	NI.NCO—Network Communication & Organization NI.C—Cybersecurity	DA.S—Storage DA.CVT—Collection, Visualization, & Transformation DA.IM—Inference & Models	AP.A—Algorithms AP.V—Variables AP.C—Control AP.M—Modularity AP.PD—Program Development	IC.C—Culture IC.SI—Social Interactions IC.SLE—Safety, Law, & Ethics

COMPUTER SCIENCE (CS) PRACTICES:

There are seven (7) CS Practices that are to be embedded in curriculum and instruction as the standards and benchmarks are taught and measured. The seven (7) CS Practices are listed below, and are more deeply explored on the next several pages. These CS Practices are also displayed on the introductory pages in front of each grade-band set of standards. For each grade-band, only the CS Practices that relate are in black text and the others are grayed so the reader can still see them as a set, but will know which ones apply to that particular set of standards.

Practice 1. Fostering an Inclusive Computing Culture

Practice 2. Collaborating Around Computing

Practice 3. Recognizing and Defining Computational Problems

Practice 4. Developing and Using Abstractions

Practice 5. Creating Computational Artifacts

Practice 6. Testing and Refining Computational Artifacts

Practice 7. Communicating About Computing

WYOMING CROSS-DISCIPLINARY CONNECTIONS

At the bottom of each standard’s page, you will find where these computer science standards tie in with other content areas, such as the following:

Math	Science
Career & Vocational Education	ELA
Social Studies	P.E.
Fine & Performing Arts	Health

These standards can be found on the WDE website at <http://edu.wyoming.gov/standards>.

INTERNATIONAL SOCIETY FOR TECHNOLOGY IN EDUCATION (ISTE) STANDARDS / WY DIGITAL LEARNING (DL) GUIDELINES

The Committee suggests educators use the following ISTE Standards for Students in their computer science curriculum, instruction, and activities, where appropriate. A committee was convened and developed the Wyoming Digital Learning Guidelines to assist educators in what education technology should be used at each grade level to best prepare students. (see Appendix B)

2016 ISTE STANDARDS FOR STUDENTS

- 1. Empowered Learner**
- 2. Digital Citizen**
- 3. Knowledge Constructor**
- 4. Innovative Designer**
- 5. Computational Thinker**
- 6. Creative Communicator**
- 7. Global Collaborator**

COMPUTER SCIENCE:

Computer Science is the study of computing principles, design, and applications (hardware & software); the creation, access, and use of

information through algorithms and problem solving, and the impact of computing on society.

COMPUTATIONAL THINKING:

Computational thinking is a necessary and meaningful 21st century skill. Computational thinking is defined as the thought processes involved in formulating a problem and expressing its solutions in such a way that a computer (human or machine) can effectively carry them out. Computational thinking develops into competencies in problem solving, critical thinking, productivity, and creativity. Over time, engaging in computational thought builds a student’s capacity to persevere, work efficiently, gain confidence, recognize and resolve ambiguity, generalize concepts, and communicate effectively. In order to adapt to global advancements in technology, students will need to use their computational thinking skills to formulate, articulate, and discuss solutions in a meaningful manner.

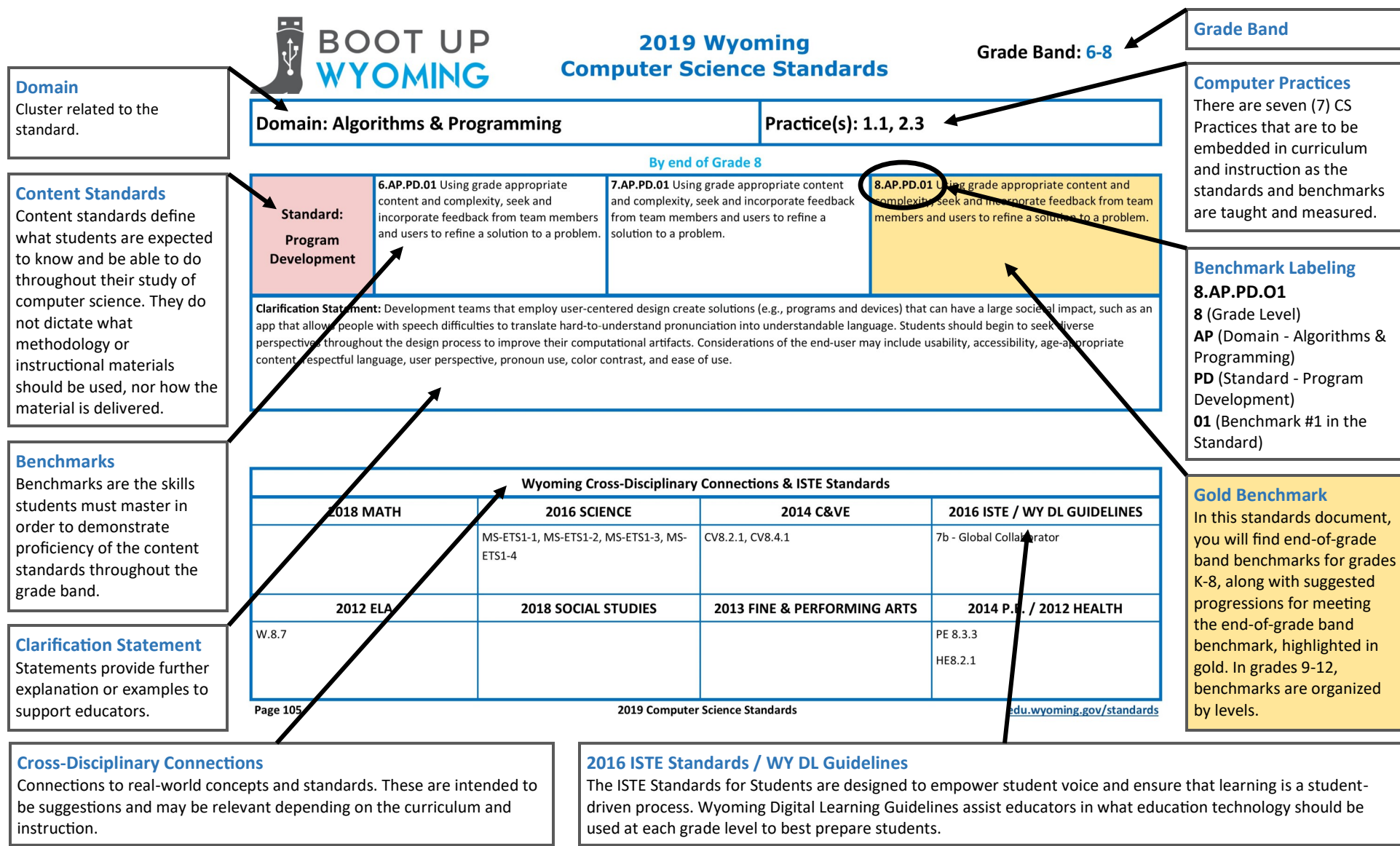
RESOURCES / REFERENCES

K-12 Computer Science Framework, (2016). Retrieved from <http://k12cs.org/>. [Ch. 5 Practices].


International Society for Technology in Education (ISTE) Standards for Students, (2016). Retrieved from <http://www.iste.org/>.

Computer Science Teachers Association (CSTA), (2017). Retrieved from <http://www.csteachers.org/page/standards>.

How to Read This Document (Grades K-8)



How to Read This Document (Grades 9-12)


2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain
Cluster related to the standard.

Domain: Algorithms & Programming	Practice(s): Level 1: 5.2; Level 2: 4.2
---	--

Grade Band

Content Standards
Content standards define what students are expected to know and be able to do throughout their study of computer science. They do not dictate what methodology or instructional materials should be used, nor how the material is delivered.

By end of Grade 12

Standard: Algorithms	<div style="display: flex;"> <div style="width: 50%; padding: 5px;"> <p>L1.AP.A.01 Create a prototype that uses algorithms (e. g., searching, sorting, finding shortest distance) to provide a possible solution for a real-world problem relevant to the student.</p> </div> <div style="width: 50%; padding: 5px;"> <p>L2.AP.A.01 Critically examine and trace classic algorithms. Use and adapt classic algorithms to solve computational problems (e.g., selection sort, insertion sort, binary search, linear search).</p> </div> </div>
Clarification Statement:	<p>Level 1: The process of developing computational artifacts embraces both creative expression and the exploration of ideas to create prototypes and solve computational problems. Students create artifacts that are personally relevant or beneficial to their community and beyond. Students should develop artifacts in response to a task or a computational problem that demonstrate the performance, reusability, and ease of implementation of an algorithm. 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.</p>

Computer Practices
There are seven (7) CS Practices that are to be embedded in curriculum and instruction as the standards and benchmarks are taught and measured.

Benchmarks
Benchmarks are the skills students must master in order to demonstrate proficiency of the content standards throughout the grade band.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
<p>L1—F.IF.A.1</p> <p>L2—F.IF.A.1, F.IF.A.3, F.IF.C.9</p>		<p>L1—CV12.3.1, CV12.4.4, CV12.5.1, CV12.5.2, CV12.5.4</p> <p>L2—CV12.4.4, CV12.5.1, CV12.5.2, CV12.5.4</p>	<p>L1—4a, 4d - Innovative Designer</p> <p>L2—4a - Innovative Designer</p>
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH

Benchmark Labeling
L2.AP.A.01
L2 (HS Level #2)
AP (Domain - Algorithms & Programming)
PD (Standard - Algorithms)
01 (Benchmark #1 in the Standard)

Clarification Statement
Statements provide further explanation or examples to support educators.

Page 145 2019 Computer Science Standards edu.wyoming.gov/standards

Gold Benchmarks
In grades 9-12, benchmarks are organized into **2 levels**. Mostly, Level 1 is intended to be at the introductory level, and Level 2 reaches at a deeper level.

Cross-Disciplinary Connections
Connections to real-world concepts and standards. These are intended to be suggestions and may be relevant depending on the curriculum and instruction.

2016 ISTE Standards / WY DL Guidelines
The ISTE Standards for Students are designed to empower student voice and ensure that learning is a student-driven process. Wyoming Digital Learning Guidelines assist educators in what education technology should be used at each grade level to best prepare students.

DESCRIPTION OF COMPUTER SCIENCE (CS) PRACTICES

CS Practice 1. Fostering an Inclusive Computing Culture

Overview: 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.

At all grade levels, students should recognize that the choices people make when they create artifacts are based on personal interests, experiences, and needs. Young learners should begin to differentiate their technology preferences from the technology preferences of others. Initially, students should be presented with perspectives from people with different backgrounds, ability levels, and points of view. As students progress, they should independently seek diverse perspectives throughout the design process for the purpose of improving their computational artifacts. Students who are well-versed in fostering an inclusive computing culture should be able to differentiate backgrounds and skill sets and know when to call upon others, such as to seek out knowledge about potential end users or intentionally seek input from people with diverse backgrounds.

1.2 Address the needs of diverse end users during the design process to produce artifacts with broad accessibility and usability.

At any level, students should recognize that users of technology have different needs and preferences and that not everyone chooses to use, or is able to use, the same technology products. For example, young learners, with teacher guidance, might compare a touchpad and a mouse to examine differences in usability. As students progress, they should consider the preferences of people

who might use their products. Students should be able to evaluate the accessibility of a product to a broad group of end users, such as people with various disabilities. For example, they may notice that allowing an end user to change font sizes and colors will make an interface usable for people with low vision. At the higher grades, students should become aware of professionally accepted accessibility standards and should be able to evaluate computational artifacts for accessibility. Students should also begin to identify potential bias during the design process to maximize accessibility in product design. For example, they can test an app and recommend to its designers that it respond to verbal commands to accommodate users who are blind or have physical disabilities.

1.3 Employ self- and peer-advocacy to address bias in interactions, product design, and development methods.

After students have experience identifying diverse perspectives and including unique perspectives (P1.1), they should begin to employ self-advocacy strategies, such as speaking for themselves if their needs are not met. As students progress, they should advocate for their peers when accommodations, such as an assistive-technology peripheral device, are needed for someone to use a computational artifact. Eventually, students should regularly advocate for both themselves and others.

CS Practice 2. Collaborating Around Computing

Overview: 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 Cultivate working relationships with individuals possessing diverse perspectives, skills, and personalities.

At any grade level, students should work collaboratively with others. Early on, they should learn strategies for working with team members who possess varying individual strengths. For example, with teacher support, students should begin to give each team member opportunities to contribute and to work with each other as co-learners. Eventually, students should become more sophisticated at applying strategies for mutual encouragement and support. They should express their ideas with logical reasoning and find ways to reconcile differences cooperatively. For example, when they disagree, they should ask others to explain their reasoning and work together to make respectful, mutual decisions. As they progress, students should use methods for giving all group members a chance to participate. Older students should strive to improve team efficiency and effectiveness by regularly evaluating group dynamics. They should use multiple strategies to make team dynamics more productive. For example, they can ask for the opinions of quieter team members, minimize interruptions by more talkative members, and give individuals credit for their ideas and their work.

2.2 Create team norms, expectations, and equitable workloads to increase efficiency and effectiveness.

After students have had experience cultivating working relationships within teams (P2.1), they should gain experience working in particular team roles. Early on, teachers may help guide this process by providing collaborative structures. For example, students may take turns in different roles on the project, such as note taker, facilitator, or “driver” of the computer. As students progress, they should become less dependent on the teacher assigning roles and become more adept at assigning roles within their teams. For example, they should decide together how to take turns in different roles. Eventually, students should independently organize their own teams and create common goals, expectations, and equitable workloads. They should also manage project workflow using agendas and timelines and should evaluate workflow to

identify areas for improvement.

2.3 Solicit and incorporate feedback from, and provide constructive feedback to, team members and other stakeholders.

At any level, students should ask questions of others and listen to their opinions. Early on, with teacher scaffolding, students should seek help and share ideas to achieve a particular purpose. As they progress in school, students should provide and receive feedback related to computing in constructive ways. For example, pair programming is a collaborative process that promotes giving and receiving feedback. Older students should engage in active listening by using questioning skills and should respond empathetically to others. As they progress, students should be able to receive feedback from multiple peers and should be able to differentiate opinions. Eventually, students should seek contributors from various environments. These contributors may include end users, experts, or general audiences from online forums.

2.4 Evaluate and select technological tools that can be used to collaborate on a project.

At any level, students should be able to use tools and methods for collaboration on a project. For example, in the early grades, students could collaboratively brainstorm by writing on a white-board. As students progress, they should use technological collaboration tools to manage team-work, such as knowledge-sharing tools and online project spaces. They should also begin to make decisions about which tools would be best to use and when to use them. Eventually, students should use different collaborative tools and methods to solicit input from not only team members and classmates but also others, such as participants in online forums or local communities.

CS Practice 3. Recognizing and Defining Computational Problems

Overview: 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.

At any level, students should be able to identify problems that have been solved computationally. For example, young students can discuss a technology that has changed the world, such as email or mobile phones. As they progress, they should ask clarifying questions to understand whether a problem or part of a problem can be solved using a computational approach. For example, identify real-world problems that span multiple disciplines, such as increasing bike safety with new helmet technology, and can be solved computationally.

3.2 Decompose complex real-world problems into manageable sub-problems that could integrate existing solutions or procedures.

At any grade level, students should be able to break problems down into their component parts. In the early grade levels, students should focus on breaking down simple problems. For example, in a visual programming environment, students could break down (or decompose) the steps needed to draw a shape. As students progress, they should decompose larger problems into manageable smaller problems. For example, young students may think of an animation as multiple scenes and thus create each scene independently. Students can also break down a program into subgoals: getting input from the user, processing the data, and displaying the result to the user. Eventually, as students encounter complex real-world problems that span multiple disciplines or social systems, they should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist. For example, students could create an app to solve a community problem that connects to an online database through an application programming interface (API).

3.3 Evaluate whether it is appropriate and feasible to solve a problem computationally.

After students have had some experience breaking problems down (P3.2) and

identifying subproblems that can be solved computationally (P3.1), they should begin to evaluate whether a computational solution is the most appropriate solution for a particular problem. For example, students might question whether using a computer to determine whether someone is telling the truth would be advantageous. As students progress, they should systematically evaluate the feasibility of using computational tools to solve given problems or subproblems, such as through a cost-benefit analysis. Eventually, students should include more factors in their evaluations, such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns.

CS Practice 4. Developing and Using Abstractions

Overview: 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.

Students at all grade levels should be able to recognize patterns. Young learners should be able to identify and describe repeated sequences in data or code through analogy to visual patterns or physical sequences of objects. As they progress, students should identify patterns as opportunities for abstraction, such as recognizing repeated patterns of code that could be more efficiently implemented as a loop. Eventually, students should extract common features from more complex phenomena or processes. For example, students should be able to identify common features in multiple segments of code and substitute a single segment that uses variables to account for the differences. In a procedure, the variables would take the form of parameters. When working with data, students should be able to identify important aspects and find patterns in related data sets such as crop output, fertilization methods, and climate conditions.

4.2 Evaluate existing technological functionalities and incorporate them into new designs.

At all levels, students should be able to use well-defined abstractions that hide complexity. Just as a car hides operating details, such as the mechanics of the engine, a computer program's "move" command relies on hidden details that cause an object to change location on the screen. As they progress, students should incorporate predefined functions into their designs, understanding that they do not need to know the underlying implementation details of the abstractions that they use. Eventually, students should understand the advantages of, and be comfortable using, existing functionalities (abstractions) including technological resources created by other people, such as libraries and application programming interfaces (APIs). Students should be able to evaluate existing abstractions to determine which should be incorporated into designs and how they should be incorporated. For example, students could build powerful apps by incorporating existing services, such as online databases that return geolocation coordinates of street names or food nutrition information.

4.3 Create modules and develop points of interaction that can apply to multiple situations and reduce complexity.

After students have had some experience identifying patterns (P4.1), decomposing problems (P3.2), using abstractions (P4.2), and taking advantage of existing resources (P4.2), they should begin to develop their own abstractions. As they progress, students should take advantage of opportunities to develop generalizable modules. For example, students could write more efficient programs by designing procedures that are used multiple times in the program. These procedures can be generalized by defining parameters that create different outputs for a wide range of inputs. Later on, students should be able to design systems of interacting modules, each with a well-defined role, that coordinate to accomplish a common goal. Within an object-oriented programming context, module design may include defining interactions among objects. At this stage, these modules, which combine both data and procedures, can be designed and documented for reuse in other

programs. Additionally, students can design points of interaction, such as a simple user interface, either text or graphical, that reduces the complexity of a solution and hides lower-level implementation details.

4.4 Model phenomena and processes and simulate systems to understand and evaluate potential outcomes.

Students at all grade levels should be able to represent patterns, processes, or phenomena. With guidance, young students can draw pictures to describe a simple pattern, such as sunrise and sunset, or show the stages in a process, such as brushing your teeth. They can also create an animation to model a phenomenon, such as evaporation. As they progress, students should understand that computers can model real-world phenomena, and they should use existing computer simulations to learn about real-world systems. For example, they may use a preprogrammed model to explore how parameters affect a system, such as how rapidly a disease spreads. Older students should model phenomena as systems, with rules governing the interactions within the system. Students should analyze and evaluate these models against real-world observations. For example, students might create a simple producer–consumer ecosystem model using a programming tool. Eventually, they could progress to creating more complex and realistic interactions between species, such as predation, competition, or symbiosis, and evaluate the model based on data gathered from nature.

CS Practice 5. Creating Computational Artifacts

Overview: 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.

At any grade level, students should participate in project planning and the creation of brainstorming documents. The youngest students may do so with the help of teachers. With scaffolding, students should gain greater independence and sophistication in the planning, design, and evaluation of artifacts. As learning progresses, students should systematically plan the development of a program or artifact and intentionally apply computational techniques, such as decomposition and abstraction, along with knowledge about existing approaches to artifact design. Students should be capable of reflecting on and, if necessary, modifying the plan to accommodate end goals.

5.2 Create a computational artifact for practical intent, personal expression, or to address a societal issue.

Students at all grade levels should develop artifacts in response to a task or a computational problem. At the earliest grade levels, students should be able to choose from a set of given commands to create simple animated stories or solve pre-existing problems. Younger students should focus on artifacts of personal importance. As they progress, student expressions should become more complex and of increasingly broader significance. Eventually, students should engage in independent, systematic use of design processes to create artifacts that solve problems with social significance by seeking input from broad audiences.

5.3 Modify an existing artifact to improve or customize it.

At all grade levels, students should be able to examine existing artifacts to understand what they do. As they progress, students should attempt to use existing solutions to accomplish a desired goal. For example, students could attach a programmable light sensor to a physical artifact they have created to make it respond to light. Later on, they should modify or remix parts of existing programs to develop something new or to add more advanced features and

complexity. For example, students could modify prewritten code from a single-player game to create a two-player game with slightly different rules.

CS Practice 6. Testing and Refining Computational Artifacts

Overview: 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 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.

At any grade level, students should be able to compare results to intended outcomes. Young students should verify whether given criteria and constraints have been met. As students progress, they should test computational artifacts by considering potential errors, such as what will happen if a user enters invalid input. Eventually, testing should become a deliberate process that is more iterative, systematic, and proactive. Older students should be able to anticipate errors and use that knowledge to drive development. For example, students can test their program with inputs associated with all potential scenarios.

6.2 Identify and fix errors using a systematic process.

At any grade level, students should be able to identify and fix errors in programs (debugging) and use strategies to solve problems with computing systems (troubleshooting). Young students could use trial and error to fix simple errors. For example, a student may try reordering the sequence of commands in a program. In a hardware context, students could try to fix a device by resetting it or checking whether it is connected to a network. As students progress, they should become more adept at debugging programs and begin to consider logic errors: cases in which a program works, but not as desired. In this way, students will examine and correct their own thinking. For

example, they might step through their program, line by line, to identify a loop that does not terminate as expected. Eventually, older students should progress to using more complex strategies for identifying and fixing errors, such as printing the value of a counter variable while a loop is running to determine how many times the loop runs.

6.3 Evaluate and refine a computational artifact multiple times to enhance its performance, reliability, usability, and accessibility.

After students have gained experience testing (P6.2), debugging, and revising (P6.1), they should begin to evaluate and refine their computational artifacts. As students progress, the process of evaluation and refinement should focus on improving performance and reliability. For example, students could observe a robot in a variety of lighting conditions to determine that a light sensor should be less sensitive. Later on, evaluation and refinement should become an iterative process that also encompasses making artifacts more usable and accessible (P1.2). For example, students can incorporate feedback from a variety of end users to help guide the size and placement of menus and buttons in a user interface.

CS Practice 7. Communicating About Computing

Overview: 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.

At any grade level, students should be able to refer to data when communicating an idea. Early on, students should, with guidance, present basic data through the use of visual representations, such as storyboards,

flowcharts, and graphs. As students progress, they should work with larger data sets and organize the data in those larger sets to make interpreting and communicating it to others easier, such as through the creation of basic data representations. Eventually, students should be able to select relevant data from large or complex data sets in support of a claim or to communicate the information in a more sophisticated manner.

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

At any grade level, students should be able to talk about choices they make while designing a computational artifact. Early on, they should use language that articulates what they are doing and identifies devices and concepts they are using with correct terminology (e.g., program, input, and debug). Younger students should identify the goals and expected outcomes of their solutions. Along the way, students should provide documentation for end users that explains their artifacts and how they function, and they should both give and receive feedback. For example, students could provide a project overview and ask for input from users. As students progress, they should incorporate clear comments in their product and document their process using text, graphics, presentations, and demonstrations.

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

All students should be able to explain the concepts of ownership and sharing. Early on, students should apply these concepts to computational ideas and creations. They should identify instances of remixing, when ideas are borrowed and iterated upon, and give proper attribution. They should also recognize the contributions of collaborators. Eventually, students should consider common licenses that place limitations or restrictions on the use of computational artifacts. For example, a downloaded image may have restrictions that prohibit modification of an image or using it for commercial purposes.

Computer Science | K-2 Introduction

K-2 Students may be most familiar with touch devices. These students may not yet understand the use of computing devices beyond playing games. They may have emerging problem-solving skills and introductory level sequencing abilities, but their understanding of programming concepts may be limited.

By the end of 2nd grade, students can:

- Select appropriate programs for appropriate tasks
- Understand the relationship between hardware and software
- Identify, deconstruct, and troubleshoot problems
- Connect and use devices and peripherals
- Begin developing keyboarding skills and utilizing other input devices
- Protect and safeguard their information
- Collect, organize, and present information through creating a computational artifact
- Organize files and analyze data
- Follow and write step-by-step instructions
- Understand that real-world circumstances can be represented using computer programs
- Understand the steps involved in the iterative process
- Understand that computer technology has positive and negative effects
- Work respectfully and responsibly with others in an online environment

WYOMING 2019 COMPUTER SCIENCE DOMAINS & STANDARDS

Computing Systems	Networks & The Internet	Data Analysis	Algorithms & Programming	Impacts of Computing
CS.D—Devices CS.HS—Hardware & Software CS.T—Troubleshooting	NI.NCO—Network Communication & Organization NI.C—Cybersecurity	DA.S—Storage DA.CVT—Collection, Visualization, & Transformation DA.IM—Inference & Models	AP.A—Algorithms AP.V—Variables AP.C—Control AP.M—Modularity AP.PD—Program Development	IC.C—Culture IC.SI—Social Interactions IC.SLE—Safety, Law, & Ethics

K-2 Computer Science Practices

There are seven (7) CS Practices that are to be embedded in curriculum and instruction as the standards and benchmarks are taught and measured. The seven (7) CS Practices are listed below, and are more deeply explored on the next several pages. For each grade-band, only the CS Practices that relate are in black text and the others are grayed so the reader can still see them as a set, but will know which ones apply to that particular set of standards.

Practice 1. Fostering an Inclusive Computing Culture

Practice 2. Collaborating Around Computing

Practice 3. Recognizing and Defining Computational Problems

Practice 4. Developing and Using Abstractions

Practice 5. Creating Computational Artifacts

Practice 6. Testing and Refining Computational Artifacts

Practice 7. Communicating About Computing

DESCRIPTION OF K-2 COMPUTER SCIENCE (CS) PRACTICES

CS Practice 1. Fostering an Inclusive Computing Culture

Overview: 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.

At all grade levels, students should recognize that the choices people make when they create artifacts are based on personal interests, experiences, and needs. Young learners should begin to differentiate their technology preferences from the technology preferences of others. Initially, students should be presented with perspectives from people with different backgrounds, ability levels, and points of view. As students progress, they should independently seek diverse perspectives throughout the design process for the purpose of improving their computational artifacts. Students who are well-versed in fostering an inclusive computing culture should be able to differentiate backgrounds and skill sets and know when to call upon others, such as to seek out knowledge about potential end users or intentionally seek input from people with diverse backgrounds.

1.2 Address the needs of diverse end users during the design process to produce artifacts with broad accessibility and usability.

At any level, students should recognize that users of technology have different needs and preferences and that not everyone chooses to use, or is able to use, the same technology products. For example, young learners, with teacher guidance, might compare a touchpad and a mouse to examine differences in usability. As students progress, they should consider the preferences of people

who might use their products. Students should be able to evaluate the accessibility of a product to a broad group of end users, such as people with various disabilities. For example, they may notice that allowing an end user to change font sizes and colors will make an interface usable for people with low vision. At the higher grades, students should become aware of professionally accepted accessibility standards and should be able to evaluate computational artifacts for accessibility. Students should also begin to identify potential bias during the design process to maximize accessibility in product design. For example, they can test an app and recommend to its designers that it respond to verbal commands to accommodate users who are blind or have physical disabilities.

1.3 Employ self- and peer-advocacy to address bias in interactions, product design, and development methods.

After students have experience identifying diverse perspectives and including unique perspectives (P1.1), they should begin to employ self-advocacy strategies, such as speaking for themselves if their needs are not met. As students progress, they should advocate for their peers when accommodations, such as an assistive-technology peripheral device, are needed for someone to use a computational artifact. Eventually, students should regularly advocate for both themselves and others.

CS Practice 2. Collaborating Around Computing

Overview: 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 Cultivate working relationships with individuals possessing diverse perspectives, skills, and personalities.

At any grade level, students should work collaboratively with others. Early on, they should learn strategies for working with team members who possess varying individual strengths. For example, with teacher support, students should begin to give each team member opportunities to contribute and to work with each other as co-learners. Eventually, students should become more sophisticated at applying strategies for mutual encouragement and support. They should express their ideas with logical reasoning and find ways to reconcile differences cooperatively. For example, when they disagree, they should ask others to explain their reasoning and work together to make respectful, mutual decisions. As they progress, students should use methods for giving all group members a chance to participate. Older students should strive to improve team efficiency and effectiveness by regularly evaluating group dynamics. They should use multiple strategies to make team dynamics more productive. For example, they can ask for the opinions of quieter team members, minimize interruptions by more talkative members, and give individuals credit for their ideas and their work.

2.2 Create team norms, expectations, and equitable workloads to increase efficiency and effectiveness.

After students have had experience cultivating working relationships within teams (P2.1), they should gain experience working in particular team roles. Early on, teachers may help guide this process by providing collaborative structures. For example, students may take turns in different roles on the project, such as note taker, facilitator, or “driver” of the computer. As students progress, they should become less dependent on the teacher assigning roles and become more adept at assigning roles within their teams. For example, they should decide together how to take turns in different roles. Eventually, students should independently organize their own teams and create common goals, expectations, and equitable workloads. They should also manage project workflow using agendas and timelines and should evaluate workflow to

identify areas for improvement.

2.3 Solicit and incorporate feedback from, and provide constructive feedback to, team members and other stakeholders.

At any level, students should ask questions of others and listen to their opinions. Early on, with teacher scaffolding, students should seek help and share ideas to achieve a particular purpose. As they progress in school, students should provide and receive feedback related to computing in constructive ways. For example, pair programming is a collaborative process that promotes giving and receiving feedback. Older students should engage in active listening by using questioning skills and should respond empathetically to others. As they progress, students should be able to receive feedback from multiple peers and should be able to differentiate opinions. Eventually, students should seek contributors from various environments. These contributors may include end users, experts, or general audiences from online forums.

2.4 Evaluate and select technological tools that can be used to collaborate on a project.

At any level, students should be able to use tools and methods for collaboration on a project. For example, in the early grades, students could collaboratively brainstorm by writing on a white-board. As students progress, they should use technological collaboration tools to manage team-work, such as knowledge-sharing tools and online project spaces. They should also begin to make decisions about which tools would be best to use and when to use them. Eventually, students should use different collaborative tools and methods to solicit input from not only team members and classmates but also others, such as participants in online forums or local communities.

CS Practice 3. Recognizing and Defining Computational Problems

Overview: 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.

At any level, students should be able to identify problems that have been solved computationally. For example, young students can discuss a technology that has changed the world, such as email or mobile phones. As they progress, they should ask clarifying questions to understand whether a problem or part of a problem can be solved using a computational approach. For example, identify real-world problems that span multiple disciplines, such as increasing bike safety with new helmet technology, and can be solved computationally.

3.2 Decompose complex real-world problems into manageable sub-problems that could integrate existing solutions or procedures.

At any grade level, students should be able to break problems down into their component parts. In the early grade levels, students should focus on breaking down simple problems. For example, in a visual programming environment, students could break down (or decompose) the steps needed to draw a shape. As students progress, they should decompose larger problems into manageable smaller problems. For example, young students may think of an animation as multiple scenes and thus create each scene independently. Students can also break down a program into subgoals: getting input from the user, processing the data, and displaying the result to the user. Eventually, as students encounter complex real-world problems that span multiple disciplines or social systems, they should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist. For example, students could create an app to solve a community problem that connects to an online database through an application programming interface (API).

3.3 Evaluate whether it is appropriate and feasible to solve a problem computationally.

After students have had some experience breaking problems down (P3.2) and

identifying subproblems that can be solved computationally (P3.1), they should begin to evaluate whether a computational solution is the most appropriate solution for a particular problem. For example, students might question whether using a computer to determine whether someone is telling the truth would be advantageous. As students progress, they should systematically evaluate the feasibility of using computational tools to solve given problems or subproblems, such as through a cost-benefit analysis. Eventually, students should include more factors in their evaluations, such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns.

CS Practice 4. Developing and Using Abstractions

Overview: 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.

Students at all grade levels should be able to recognize patterns. Young learners should be able to identify and describe repeated sequences in data or code through analogy to visual patterns or physical sequences of objects. As they progress, students should identify patterns as opportunities for abstraction, such as recognizing repeated patterns of code that could be more efficiently implemented as a loop. Eventually, students should extract common features from more complex phenomena or processes. For example, students should be able to identify common features in multiple segments of code and substitute a single segment that uses variables to account for the differences. In a procedure, the variables would take the form of parameters. When working with data, students should be able to identify important aspects and find patterns in related data sets such as crop output, fertilization methods, and climate conditions.

4.2 Evaluate existing technological functionalities and incorporate them into new designs.

At all levels, students should be able to use well-defined abstractions that hide complexity. Just as a car hides operating details, such as the mechanics of the engine, a computer program's "move" command relies on hidden details that cause an object to change location on the screen. As they progress, students should incorporate predefined functions into their designs, understanding that they do not need to know the underlying implementation details of the abstractions that they use. Eventually, students should understand the advantages of, and be comfortable using, existing functionalities (abstractions) including technological resources created by other people, such as libraries and application programming interfaces (APIs). Students should be able to evaluate existing abstractions to determine which should be incorporated into designs and how they should be incorporated. For example, students could build powerful apps by incorporating existing services, such as online databases that return geolocation coordinates of street names or food nutrition information.

4.3 Create modules and develop points of interaction that can apply to multiple situations and reduce complexity.

After students have had some experience identifying patterns (P4.1), decomposing problems (P3.2), using abstractions (P4.2), and taking advantage of existing resources (P4.2), they should begin to develop their own abstractions. As they progress, students should take advantage of opportunities to develop generalizable modules. For example, students could write more efficient programs by designing procedures that are used multiple times in the program. These procedures can be generalized by defining parameters that create different outputs for a wide range of inputs. Later on, students should be able to design systems of interacting modules, each with a well-defined role, that coordinate to accomplish a common goal. Within an object-oriented programming context, module design may include defining interactions among objects. At this stage, these modules, which combine both data and procedures, can be designed and documented for reuse in other

programs. Additionally, students can design points of interaction, such as a simple user interface, either text or graphical, that reduces the complexity of a solution and hides lower-level implementation details.

4.4 Model phenomena and processes and simulate systems to understand and evaluate potential outcomes.

Students at all grade levels should be able to represent patterns, processes, or phenomena. With guidance, young students can draw pictures to describe a simple pattern, such as sunrise and sunset, or show the stages in a process, such as brushing your teeth. They can also create an animation to model a phenomenon, such as evaporation. As they progress, students should understand that computers can model real-world phenomena, and they should use existing computer simulations to learn about real-world systems. For example, they may use a preprogrammed model to explore how parameters affect a system, such as how rapidly a disease spreads. Older students should model phenomena as systems, with rules governing the interactions within the system. Students should analyze and evaluate these models against real-world observations. For example, students might create a simple producer–consumer ecosystem model using a programming tool. Eventually, they could progress to creating more complex and realistic interactions between species, such as predation, competition, or symbiosis, and evaluate the model based on data gathered from nature.

CS Practice 5. Creating Computational Artifacts

Overview: 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.

At any grade level, students should participate in project planning and the creation of brainstorming documents. The youngest students may do so with the help of teachers. With scaffolding, students should gain greater independence and sophistication in the planning, design, and evaluation of artifacts. As learning progresses, students should systematically plan the development of a program or artifact and intentionally apply computational techniques, such as decomposition and abstraction, along with knowledge about existing approaches to artifact design. Students should be capable of reflecting on and, if necessary, modifying the plan to accommodate end goals.

5.2 Create a computational artifact for practical intent, personal expression, or to address a societal issue.

Students at all grade levels should develop artifacts in response to a task or a computational problem. At the earliest grade levels, students should be able to choose from a set of given commands to create simple animated stories or solve pre-existing problems. Younger students should focus on artifacts of personal importance. As they progress, student expressions should become more complex and of increasingly broader significance. Eventually, students should engage in independent, systematic use of design processes to create artifacts that solve problems with social significance by seeking input from broad audiences.

5.3 Modify an existing artifact to improve or customize it.

At all grade levels, students should be able to examine existing artifacts to understand what they do. As they progress, students should attempt to use existing solutions to accomplish a desired goal. For example, students could attach a programmable light sensor to a physical artifact they have created to make it respond to light. Later on, they should modify or remix parts of existing programs to develop something new or to add more advanced features and

complexity. For example, students could modify prewritten code from a single-player game to create a two-player game with slightly different rules.

CS Practice 6. Testing and Refining Computational Artifacts

Overview: 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 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.

At any grade level, students should be able to compare results to intended outcomes. Young students should verify whether given criteria and constraints have been met. As students progress, they should test computational artifacts by considering potential errors, such as what will happen if a user enters invalid input. Eventually, testing should become a deliberate process that is more iterative, systematic, and proactive. Older students should be able to anticipate errors and use that knowledge to drive development. For example, students can test their program with inputs associated with all potential scenarios.

6.2 Identify and fix errors using a systematic process.

At any grade level, students should be able to identify and fix errors in programs (debugging) and use strategies to solve problems with computing systems (troubleshooting). Young students could use trial and error to fix simple errors. For example, a student may try reordering the sequence of commands in a program. In a hardware context, students could try to fix a device by resetting it or checking whether it is connected to a network. As students progress, they should become more adept at debugging programs and begin to consider logic errors: cases in which a program works, but not as desired. In this way, students will examine and correct their own thinking. For

example, they might step through their program, line by line, to identify a loop that does not terminate as expected. Eventually, older students should progress to using more complex strategies for identifying and fixing errors, such as printing the value of a counter variable while a loop is running to determine how many times the loop runs.

6.3 Evaluate and refine a computational artifact multiple times to enhance its performance, reliability, usability, and accessibility.

After students have gained experience testing (P6.2), debugging, and revising (P6.1), they should begin to evaluate and refine their computational artifacts. As students progress, the process of evaluation and refinement should focus on improving performance and reliability. For example, students could observe a robot in a variety of lighting conditions to determine that a light sensor should be less sensitive. Later on, evaluation and refinement should become an iterative process that also encompasses making artifacts more usable and accessible (P1.2). For example, students can incorporate feedback from a variety of end users to help guide the size and placement of menus and buttons in a user interface.

CS Practice 7. Communicating About Computing

Overview: 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.

At any grade level, students should be able to refer to data when communicating an idea. Early on, students should, with guidance, present basic data through the use of visual representations, such as storyboards,

flowcharts, and graphs. As students progress, they should work with larger data sets and organize the data in those larger sets to make interpreting and communicating it to others easier, such as through the creation of basic data representations. Eventually, students should be able to select relevant data from large or complex data sets in support of a claim or to communicate the information in a more sophisticated manner.

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

At any grade level, students should be able to talk about choices they make while designing a computational artifact. Early on, they should use language that articulates what they are doing and identifies devices and concepts they are using with correct terminology (e.g., program, input, and debug). Younger students should identify the goals and expected outcomes of their solutions. Along the way, students should provide documentation for end users that explains their artifacts and how they function, and they should both give and receive feedback. For example, students could provide a project overview and ask for input from users. As students progress, they should incorporate clear comments in their product and document their process using text, graphics, presentations, and demonstrations.

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

All students should be able to explain the concepts of ownership and sharing. Early on, students should apply these concepts to computational ideas and creations. They should identify instances of remixing, when ideas are borrowed and iterated upon, and give proper attribution. They should also recognize the contributions of collaborators. Eventually, students should consider common licenses that place limitations or restrictions on the use of computational artifacts. For example, a downloaded image may have restrictions that prohibit modification of an image or using it for commercial purposes.



2019 Wyoming Computer Science Standards

Grade Band: **K-2**

Domain: Computing Systems

Practice(s): 1.1

By end of Grade 2

Standard: Devices	K.CS.D.01 With guidance, follow directions and start to make appropriate choices to use computing devices to perform a variety of tasks (e.g., turn on, select, open and close programs, logon and logoff).	1.CS.D.01 With guidance, select and use a computing device to perform a variety of tasks for an intended outcome (e.g., turn on, select, open and close programs, logon and logoff).	2.CS.D.01 Independently select and use a computing device to perform a variety of tasks for an intended outcome (e.g., create an artifact).
------------------------------	--	---	--

Clarification Statement: People use computing devices to perform a variety of tasks accurately and quickly. Students should be able to select the appropriate app/program to use for tasks they are required to complete. For example, if students are asked to draw a picture, they should be able to open and use a drawing app/program to complete this task, or if they are asked to create a presentation, they should be able to open and use presentation software. In addition, with teacher guidance, students should compare and discuss preferences for software with the same primary functionality. Students could compare different programs (e.g., web browsers, word processing, presentation, or drawing).

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	K-2-ETS1-3		1d - Empowered Learner
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
	SS2.6.3	FPA4.1.D.6	



Domain: Computing Systems

Practice(s): 7.2

By end of Grade 2

Standard: Hardware & Software

K.CS.HS.01 Use appropriate terminology to identify and use common computing devices, components, and software in a variety of environments (e.g., desktop computer, laptop computer, tablet device, monitor, keyboard, mouse, or printer).

1.CS.HS.01 Use appropriate terminology in naming and demonstrate the function of common computing devices, components, and software (e.g., use of a printer, appropriate input device use, or common operating system features).

2.CS.HS.01 Demonstrate and describe the function of common components of computing systems (hardware and software) (e.g. use a browser, search engine).

Clarification Statement: A computing system is composed of hardware and software. Hardware consists of physical components. Software consists of the programs and applications that run on the hardware. 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. Students should understand the relationship between hardware and software. Software consists of the programs that give the hardware useful functionality.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
			1d - Empowered Learner
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
W.2.2, L.2.1, L.2.2, L.2.3, L.2.4, L.2.5, L.2.6			



2019 Wyoming Computer Science Standards

Grade Band: **K-2**

Domain: Computing Systems

Practice(s): 6.2, 7.2

By end of Grade 2

Standard: Troubleshooting	K.CS.T.01 Recognize computing systems might not work as expected and identify and effectively communicate simple hardware or software problems. Implement solutions with guidance (e.g., volume turned down on headphones, monitor turned off).	1.CS.T.01 Recognize computing systems might not work as expected and identify and effectively communicate simple hardware or software problems. Implement solutions with guidance (e.g., app or program is not working as expected, no sound is coming from the device, caps lock turned on).	2.CS.T.01 Recognize computing systems might not work as expected and identify and effectively communicate simple hardware or software problems. Implement solutions with guidance (e.g., app or program is not working as expected, no sound is coming from the device, caps lock turned on) and discuss problems with peers and adults.
<p>Clarification Statement: Problems with computing systems have different causes. Students at this level do not need to understand those causes, but they should be able to communicate a problem with accurate terminology (e.g., when an app or program is not working as expected, a device will not turn on, the sound does not work, etc.). Ideally, students would be able to use simple troubleshooting strategies, including turning a device off and on to reboot it, closing and reopening an app, turning on speakers, or plugging in headphones.</p>			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV5.3.3, CV5.4.3	1c, 1d - Empowered Learner
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
L.2.3, L.2.6			HE2.2.2



2019 Wyoming Computer Science Standards

Grade Band: **K-2**

Domain: Networks & the Internet

Practice(s): 6.2

By end of Grade 2

Standard: Network Communication & Organization	K.NI.NCO.01 Recognize and discuss that computing devices can be connected together.	1.NI.NCO.01 Identify and describe that by connecting computing devices together they can share information (e.g., remote storage, printing, the internet).	2.NI.NCO.01 Identify and describe that computing devices can be connected in a variety of ways (e.g., Bluetooth, Wi-Fi, home and school networks, the internet).

Clarification Statement: Computing devices are connected in a variety of ways. Students at this level need to understand that connectivity is part of the overall computing environment and that different protocols (e.g., wired, wireless, Wi-Fi, Bluetooth) are used depending on the device purpose.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV5.4.3	1d - Empowered Learner
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
W.2.2, L.2.1, L.2.2, L.2.3, L.2.4, L.2.5, L.2.6			



2019 Wyoming Computer Science Standards

Grade Band: **K-2**

Domain: Networks & the Internet

Practice(s): 7.3

By end of Grade 2

Standard: Cybersecurity	K.NI.C.01 Discuss what authentication factors are and why we do not share them with others. With guidance, use them to access technological devices, apps, etc.	1.NI.C.01 Identify what authentication factors are, explain why they are not shared, and discuss what makes authentication effective. Independently use them to access technological devices, apps, etc.	2.NI.C.01 Explain what authentication factors are, why we use them, and apply authentication to protect devices and information (personal and private) from unauthorized access.
Clarification Statement: Learning to protect one's device or information from unwanted use by others is an essential first step in learning about cybersecurity. Students should appropriately use and protect the authentication methods that are required.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV5.4.3	2d - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
W.2.2, L.2.1, L.2.2, L.2.3, L.2.4, L.2.5, L.2.6			



2019 Wyoming Computer Science Standards

Grade Band: **K-2**

Domain: Data Analysis

Practice(s): 4.2

By end of Grade 2

**Standard:
Storage**

K.DA.S.01 With guidance, locate, open, modify and save an existing file with a computing device.

1.DA.S.01 With guidance, locate, open, modify and save an existing file, and use appropriate file-naming conventions. Recognize that the file exists within an organizational structure (drive, folder, file).

2.DA.S.01 With guidance, develop and modify an organizational structure by creating, copying, moving, and deleting files and folders.

Clarification Statement: All information stored and processed by a computing device is referred to as data. Data can be images, text documents, audio files, software programs or apps, video files, etc. As students use software to complete tasks on a computing device, they will be manipulating data in files. They will be organizing that information in folders and a file structure.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV5.1.4	1d - Empowered Learner
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: **K-2**

Domain: Data Analysis

Practice(s): 4.4, 7.1

By end of Grade 2

Standard: Collection, Visualization, & Transformation	K.DA.CVT.01 With guidance, collect data and present it visually.	1.DA.CVT.01 With guidance, collect data and present it in more than one way (e.g. written and visual presentation).	2.DA.CVT.01 With guidance, collect data and independently present the same data in various visual formats.
<p>Clarification Statement: The collection and use of data about the world around them is a routine part of life and influences how people live. Students could collect data on the weather, such as sunny days versus rainy days, the temperature at the beginning of the school day and end of the school day, or the inches of rain over the course of a storm. Students could count the number of pieces of each color of candy in a bag of candy, such as Skittles or M&Ms. Students could create surveys of things that interest them, such as favorite foods, pets, or TV shows, and collect answers to their surveys from their peers and others. The data collected could then be organized into two or more visualizations, such as a bar graph, pie chart, or pictograph.</p>			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
1.MD.J.4, 2.MD.I.10a	K-2-ETS1-3, K-PS2-1, K-PS2-2, K-PS3-1, K-LS1-1, K-ESS2-1, K-ESS2-2	CV5.4.1	3c - Knowledge Constructor
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
	SS2.5.1		

Domain: Data Analysis
Practice(s): 4.1
By end of Grade 2

Standard: Inference & Models	K.DA.IM.01 With guidance, draw conclusions and make predictions based on picture graphs or patterns with or without a computing device (e.g., make predictions based on weather data presented in a picture graph or complete a pattern).	1.DA.IM.01 With guidance, identify and interpret data from a chart or graph (visualization) in order to make a prediction, with or without a computing device.	2.DA.IM.01 With guidance, interpret data and present it in a chart or graph (visualization) in order to make a prediction, with or without a computing device.
---	--	---	---

Clarification Statement: Data can be used to make inferences or predictions about the world. Students could analyze a graph or pie chart of the colors in a bag of candy or the averages for colors in multiple bags of candy, identify the patterns for which colors are most and least represented, and then make a prediction as to which colors will have most and least in a new bag of candy. Students could analyze graphs of temperatures taken at the beginning of the school day and end of the school day, identify the patterns of when temperatures rise and fall, and predict if they think the temperature will rise or fall at a particular time of the day, based on the patterns observed.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
1.MD.J.4, 2.MD.I.10a	K-2-ETS1-3, K-PS2-1, K-PS3-1, K-LS1-1, K-ESS3-3	CV5.4.1, CV5.4.4	6c - Creative Communicator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
RI.2.7	SS2.5.1		



Domain: Algorithms & Programming

Practice(s): 4.4

By end of Grade 2

Standard: Algorithms	K.AP.A.01 With guidance, model daily processes and follow algorithms (sets of step-by-step instructions) to complete tasks (e.g., verbally, kinesthetically, with robot devices, or a programming language).	1.AP.A.01 With guidance, identify and model daily processes and follow algorithms (sets of step-by-step instructions) to complete tasks (e.g., verbally, kinesthetically, with robot devices, or a programming language).	2.AP.A.01 With guidance, identify and model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks (e.g., verbally, kinesthetically, with robot devices, or a programming language).
Clarification Statement: Students model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks. Students could create and follow algorithms for making simple foods, brushing their teeth, getting ready for school, or participating in clean-up time.			

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
K.G.H.1, 2.G.J.3a, 2.G.J.3b, 2.G.J.3c, 2.NBT.E.9	K-2-ETS1-1, K-LS1-1, K-ESS3-3	CV5.4.1	4a - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
RI.2.3, W.2.6, SL.2.5, RI.2.7		FPA4.1.A.1, FPA4.1.A.2, FPA4.1.D.5, FPA4.1.M.4, FPA4.1.T.1, FPA4.1.T.2	

Domain: Algorithms & Programming
Practice(s): 4.1
By end of Grade 2

Standard: Variables	K.AP.V.01 With guidance, demonstrate that data may be represented by symbols (e.g., thumbs up/down as representations of yes/no, arrows when writing algorithms to represent direction, or encode and decode words using numbers, pictographs, or other symbols to represent letters or words).	1.AP.V.01 With guidance, demonstrate that computers represent data using numbers, letters, words, or other symbols (e.g., thumbs up/down as representations of yes/no, arrows when writing algorithms to represent direction, or encode and decode words using numbers, pictographs, or other symbols to represent letters or words).	2.AP.V.01 Model the way programs store and manipulate data by using numbers or other symbols to represent information (e.g., thumbs up/down as representations of yes/no, arrows when writing algorithms to represent direction, or encode and decode words using numbers, pictographs, or other symbols to represent letters or words).
--------------------------------	--	--	---

Clarification Statement: Information in the real world can be represented in computer programs. Students could use thumbs up/down as representations of yes/no, use arrows when writing algorithms to represent direction, or encode and decode words using numbers, pictographs, or other symbols to represent letters or words.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
K.G.H.1, K.CC.B.4, K.CC.B.4a, K.CC.B.4b, K.CC.B.4c, 1.MD.J.4, 1.OA.A.1, 1.OA.C.5, 1.OA.C.6, 1.G.K.1, 1.G.K.2, 2.G.J.2, 2.OA.A.1, 2.G.J.3a, 2.G.J.3b, 2.G.J.3c	K-PS3-1, K-LS1-1, K-ESS3-1, K-ESS3-3		
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
RI.2.7			

Domain: Algorithms & Programming
Practice(s): 5.2

By end of Grade 2

Standard: Control	K.AP.C.01 With guidance, independently or collaboratively create programs to accomplish tasks using sequencing (emphasizing the beginning, middle, and end).	1.AP.C.01 With guidance, independently or collaboratively create programs to accomplish tasks using sequencing, conditionals, and repetition (e.g., program a robot device, or algorithmically describe an unplugged activity).	2.AP.C.01 With guidance, independently and collaboratively create programs to accomplish tasks using a programming language, robot device, or unplugged activity that includes sequencing, conditionals, and repetition.
------------------------------	---	--	---

Clarification Statement: 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 an animation of a butterfly, a loop could be combined with move commands to allow continual but controlled movement of the character.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
K.G.H.1, 1.MD.J.4, 1.OA.A.1, 1.G.K.1, 1.G.K.2, 2.OA.A.1	K-2-ETS1-3, K-PS2-1, K-PS2-2	CV5.4.1	4a - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
RI.2.3, W.2.6, SL.2.5		FPA4.1.A.1, FPA4.1.A.2, FPA4.1.D.5, FPA4.1.M.4, FPA4.1.T.1, FPA4.1.T.2	

Domain: Algorithms & Programming
Practice(s): 3.2

By end of Grade 2

Standard: Modularity	K.AP.M.01 Using grade appropriate content and complexity, decompose (breakdown) the steps needed to solve a problem into a precise sequence of instructions (e.g., to show the life cycle of a plant - plant seed in dirt, water dirt, plant begins to grow with sunlight).	1.AP.M.01 Using grade appropriate content and complexity, decompose (breakdown) the steps needed to solve a problem into a precise sequence of instructions (e.g., given a deck of cards, have students sort them by multiple methods - color, suit, or rank).	2.AP.M.01 Using grade appropriate content and complexity, decompose (breakdown) the steps needed to solve a problem into a precise sequence of instructions (e.g., develop a set of instructions on how to play your favorite game).
Clarification Statement: Learning to program in modules first involves learning to break problems into steps. Decomposition is the act of breaking down tasks into simpler tasks. Students could break down the steps needed to make a peanut butter and jelly sandwich, to brush their teeth, to draw a shape, to move a character across the screen, or to solve a level of a coding app.			

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
K.G.I.5, K.MD.F.1, K.CC.B.4, K.CC.B.4a, K.CC.B.4b, K.CC.B.4c, 1.MD.H.2, 1.MD.J.4, 1.OA.A.1, 1.OA.C.5, 1.OA.C.6, 1.G.K.1, 1.G.K.2, 2.MD.F.1, 2.G.J.2, 2.G.J.3a, 2.G.J.3b, 2.G.J.3c, 2.OA.A.1	K-2-ETS1-1, K-2-ETS1-2, K-PS3-2	CV5.4.3	5c - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
W.2.5		FPA4.1.A.3	



Domain: Algorithms & Programming

Practice(s): 5.1, 7.2

By end of Grade 2

Standard: Program Development	K.AP.PD.01 With guidance, develop plans that describe a program's sequence of events, goals, and expected outcomes.	1.AP.PD.01 Independently or with guidance, develop plans that describe a program's sequence of events, goals, and expected outcomes.	2.AP.PD.01 Develop plans that describe a program's sequence of events, goals, and expected outcomes.
Clarification Statement: Creating a plan for what a program will do clarifies the steps that will be needed to create a program and can be used to check if a program is correct. Students could create a planning document, such as a story map, a storyboard, or a sequential graphic organizer, to illustrate what their program will do. Students at this stage may complete the planning process with help from their teachers.			

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	K-2-ETS1-1, K-PS2-1, K-PS3-2, K-ESS3-3	CV5.4.3	4a - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
W.2.2		FPA4.1.A.3, FPA4.1.T.5	



2019 Wyoming Computer Science Standards

Grade Band: **K-2**

Domain: Algorithms & Programming

Practice(s): 7.3

By end of Grade 2

Standard: Program Development	K.AP.PD.02 Independently or with guidance, give credit to ideas, creations, and solutions of others while developing algorithms.	1.AP.PD.02 Independently or with guidance, give credit to ideas, creations, and solutions of others while writing and/or developing programs.	2.AP.PD.02 Give credit to ideas, creations, and solutions of others while writing and developing programs.
Clarification Statement: Using computers comes with a level of responsibility. Students should credit artifacts that were created by others, such as pictures, music, and code. Credit could be given orally, if presenting their work to the class, or in writing or orally, if sharing work on a class blog or website. Proper attribution at this stage does not require a formal citation, such as in a bibliography or works cited document.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	K-ESS2-2, K-ESS3-1		2c - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
W.2.2, W.2.5			



Domain: Algorithms & Programming

Practice(s): 6.2

By end of Grade 2

Standard: Program Development	K.AP.PD.03 With guidance, independently or collaboratively debug (identify and fix errors) algorithms using a programming language and/or unplugged activity.	1.AP.PD.03 With guidance, independently or collaboratively debug (identify and fix errors) programs using a programming language and/or unplugged activity.	2.AP.PD.03 Independently and collaboratively debug (identify and fix errors) programs using a programming language.
--	--	--	--

Clarification Statement: Algorithms or programs may not always work correctly. Students should be able to use various strategies, such as changing the sequence of the steps, following the algorithm in a step-by-step manner, or trial and error to fix problems in algorithms and programs.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
K.G.H.1, 1.MD.J.4, 1.OA.A.1, 1.G.K.1, 2.G.J.2, 2.OA.A.1		CV5.4.2, CV5.4.3	4c - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



Domain: Algorithms & Programming

Practice(s): 7.2

By end of Grade 2

Standard: Program Development	K.AP.PD.04 Use correct terminology (beginning, middle, end) in the development of an algorithm.	1.AP.PD.04 Use correct terminology (first step, second step, third step) and explain the choices made in the development of an algorithm.	2.AP.PD.04 Use correct terminology (debug, program input/output, code) to explain the development of a program or an algorithm (e.g., in an unplugged activity, hands on manipulatives, or a programming language).
--	--	--	--

Clarification Statement: At this stage, students should be able to talk or write about the goals and expected outcomes of the programs they create and the choices that they made when creating programs. This could be done using coding journals, discussions with a teacher, class presentations, or blogs.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV5.4.2, CV5.4.3	3d - Knowledge Constructor
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
L.2.4, L.2.6			



2019 Wyoming Computer Science Standards

Grade Band: **K-2**

Domain: Impacts of Computing

Practice(s): 3.1

By end of Grade 2

**Standard:
Culture**

K.IC.C.01 Discuss different ways in which technology is used in your daily life.

1.IC.C.01 Identify how people use different types of technologies in their daily work and personal lives.

2.IC.C.01 Describe how people use different types of technologies in their daily work and personal lives.

Clarification Statement: 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 greater accessibility.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	K-2-ETS1-1, K-ESS3-2, K-ESS3-3		
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
W.2.2, L.2.1, L.2.2, L.2.3, L.2.4, L.2.5, L.2.6	SS2.3.3, SS2.4.2		HE2.4.8



2019 Wyoming Computer Science Standards

Grade Band: **K-2**

Domain: Impacts of Computing

Practice(s): 2.1

By end of Grade 2

Standard: Social Interactions	K.IC.SI.01 With guidance, identify appropriate manners while participating in an online environment.	1.IC.SI.01 With guidance, identify appropriate and inappropriate behavior. Act responsibly while participating in an online community and know how to report concerns.	2.IC.SI.01 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.
Clarification Statement: Online communication facilitates positive interactions, such as sharing ideas with many people, but the public and anonymous nature of online communication also allows intimidating and inappropriate behavior in the form of cyberbullying. Students could share their work on blogs or in other collaborative spaces online, taking care to avoid sharing information that is inappropriate or that could personally identify them to others. Students could provide feedback to others on their work in a kind and respectful manner and could tell an adult if others are sharing things they should not share or are treating others in an unkind or disrespectful manner on online collaborative spaces.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV5.2.3, CV5.2.4, CV5.5.3, CV5.5.4	2b - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
SL.2.1.a		FPA4.1.A.5, FPA4.4.M.1, FPA4.4.T.2	PE 2.3.1 HE2.3.3

Performance Level Descriptors (PLDs)

Grade Band: K-2

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Devices: 2.CS.D.01 Independently select and use a computing device to perform a variety of tasks for an intended outcome (e.g., create an artifact).	provides little to no evidence in addressing the expectation(s).	with guidance, uses a computing device to complete assignments or teacher led activities.	regularly uses a computing device to independently <ul style="list-style-type: none"> - power on and off devices. - authenticate, when applicable. - open appropriate programs. - complete assignments or teacher led activities. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., can recognize capabilities of multiple devices and can perform similar tasks with them).
Hardware & Software: 2.CS.HS.01 Demonstrate and describe the function of common components of computing systems (hardware and software) (e.g. use a browser, search engine).	provides little to no evidence in addressing the expectation(s).	with guidance: <ul style="list-style-type: none"> - identifies hardware components and software applications. - utilizes hardware components and software applications. 	can identify and utilize: <ul style="list-style-type: none"> - a variety of hardware components (e.g., input devices, printers). - software applications (e.g., browsers, apps). - navigation to browser search engines and applications. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., justifies hardware and software choices).
Troubleshooting: 2.CS.T.01 Recognize that computing systems might not work as expected and identify and effectively communicate simple hardware or software problems and implement solutions (e.g., app or program is not working as expected, no sound is coming from the device, caps lock turned on) and discuss problems with peers and adults.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - can recognize that computing systems may not work as expected. - with guidance, identifies and effectively communicates simple hardware and software problems. - with guidance, implements solutions to simple hardware or software issues. 	<ul style="list-style-type: none"> - can recognize that computing systems may not work as expected. - identifies and effectively communicates simple hardware and software problems. - implements solutions to simple hardware or software issues. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., helps others in troubleshooting issues, can troubleshoot more complex issues like connectivity or advanced software features).

Performance Level Descriptors (PLDs)

Grade Band: K-2

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Network Communication & Organization: 2.NI.NCO.01 Identify and describe that computing devices can be connected in a variety of ways (e.g., Bluetooth, Wi-Fi, home and school networks, the internet).	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - can identify that computing devices can be connected in a variety of ways. - with guidance, can describe a connectivity option (e.g., Wi-Fi or Bluetooth). 	<ul style="list-style-type: none"> - can identify that computing devices can be connected in a variety of ways. - can describe different connectivity options (e.g., Bluetooth, internet). 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., evaluates the appropriateness of different connectivity options for a variety of tasks).
Cybersecurity: 2.NI.C.01 Explain what authentication factors are, why we use them, and apply authentication to protect devices and information (personal and private) from unauthorized access.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - identifies what authentication factors are. - with guidance, applies authentication factors to appropriate apps and devices. 	<ul style="list-style-type: none"> - explains what authentication factors are and why we use them. - applies authentication factors to appropriate apps and devices. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., can compare authentication methods, one factor versus two factors).
Storage: 2.DA.S.01 With guidance, develop and modify an organizational structure by creating, copying, moving, and deleting files and folders.	provides little to no evidence in addressing the expectation(s).	while working with a computing device and with guidance: <ul style="list-style-type: none"> - locates existing files. - opens existing files. - modifies existing files. - saves changes to a file. 	with guidance, develops and modifies an organizational structure by: <ul style="list-style-type: none"> - creating folders. - copying existing folders and files. - moving existing folders and files. - deleting folders and files. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., can independently create organizational structure).

Performance Level Descriptors (PLDs)

Grade Band: K-2

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Collection, Visualization, & Transformation: 2.DA.CVT.01 With guidance, collect data and independently present the same data in various visual formats.	provides little to no evidence in addressing the expectation(s).	with guidance: - creates a data set, and - presents that data.	- with guidance, creates a data set, and - independently presents that data in multiple formats (e.g., as a table and graph or as a table and chart).	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., independently creates and presents their own data sets).
Inference & Models: 2.DA.IM.01 With guidance, interpret data and present it in a chart or graph (visualization) in order to make a prediction, with or without a computing device.	provides little to no evidence in addressing the expectation(s).	with guidance: - interprets data, and - presents it in a chart or graph (visualization).	with guidance: - interprets data. - presents it in a chart or graph (visualization). - makes a prediction based on the data.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., independently perform any of the proficient student steps).
Algorithms: 2.AP.A.01 With guidance, identify and model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks (e.g., verbally, kinesthetically, with robot devices, or a programming language).	provides little to no evidence in addressing the expectation(s).	with guidance: - follows algorithms to complete tasks.	with guidance: - follows algorithms to complete tasks. - creates algorithms to complete tasks.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., independently creates algorithms via one or more of the following techniques: verbally, kinesthetically, with robot devices, or a programming language).

Performance Level Descriptors (PLDs)

Grade Band: K-2

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Variables: 2.AP.V.01 Model the way programs store and manipulate data by using numbers or other symbols to represent information (e.g. thumbs up/down as representations of yes/no, arrows when writing algorithms to represent direction, or encode and decode words using numbers, pictographs, or other symbols to represent letters or words).	provides little to no evidence in addressing the expectation(s).	with guidance: - uses symbols to represent information. - understands that inferred meanings of the symbols can change or can represent missing information. - creates expressions with symbols to convey data, information, or processes.	- uses symbols to represent information. - understands that inferred meanings of the symbols can change or can represent missing information. - creates expressions with symbols to convey data, information, or processes.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., creates complex expressions with symbols).
Control: 2.AP.C.01 With guidance, independently and collaboratively create programs to accomplish tasks using a programming language, robot device, or unplugged activity that includes sequencing, conditionals, and repetition.	provides little to no evidence in addressing the expectation(s).	with guidance, create: - programs that include sequencing, conditionals, or repetition. - tasks that include sequencing, conditionals, or repetition.	with guidance: - individually create programs or tasks that include sequencing, conditionals, and repetition. - collaboratively create programs that include sequencing, conditionals, and repetition.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., independently creates programs that include sequencing, conditionals, and repetition).
Modularity: 2.AP.M.01 Using grade appropriate content and complexity, decompose (breakdown) the steps needed to solve a problem into a precise sequence of instructions (e.g., develop a set of instructions on how to play your favorite game).	provides little to no evidence in addressing the expectation(s).	with guidance: - decomposes a problem. - creates a precise sequence of instructions to solve that problem.	- decomposes a problem. - creates a precise sequence of instructions to solve that problem.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., can create different instruction sets that accomplish the same task).

Performance Level Descriptors (PLDs)

Grade Band: K-2

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
<p>Program Development: 2.AP.PD.01 Develop plans that describe a program's sequence of events, goals, and expected outcomes.</p> <p>Program Development: 2.AP.PD.02 Give credit to ideas, creations, and solutions of others while writing and developing programs.</p> <p>Program Development: 2.AP.PD.03 Independently and collaboratively debug (identify and fix errors) programs using a programming language.</p> <p>Program Development: 2.AP.PD.04 Use correct terminology (debug, program input/output, code) to explain the development of a program or an algorithm (e.g., in an unplugged activity, hands on manipulatives, or a programming language).</p>	<p>provides little to no evidence in addressing the expectation(s).</p>	<p>with guidance, demonstrates program development by:</p> <ul style="list-style-type: none"> - creating a plan for a program. - writing the program. - giving credit for the resources used. - debugging the program. 	<p>demonstrates program development by:</p> <ul style="list-style-type: none"> - creating a plan for a program. - writing the program. - giving credit for the resources used. - debugging the program. 	<p>demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., demonstrates the development process on different platforms, languages, or mediums).</p>

Performance Level Descriptors (PLDs)

Grade Band: K-2

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Culture: 2.IC.C.01 Describe how people use different types of technologies in their daily work and personal lives.	provides little to no evidence in addressing the expectation(s).	identifies how people use different types of technologies (e.g., cell phones, computers) in their daily work and personal lives.	describes how people use different types of technologies (e.g., cell phones, computers) in their daily work and personal lives.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., identifies and describes the potential impacts of different technologies).
Social Interactions: 2.IC.SI.01 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.	provides little to no evidence in addressing the expectation(s).	with guidance: - makes appropriate choices when participating in an online community. - identifies inappropriate behavior and reporting procedures.	- makes appropriate choices when participating in an online community. - identifies inappropriate behavior and reporting procedures.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Computer Science | 3-5 Introduction

Throughout grades 3-5, students engage in creative applications of Computer Science concepts and practices introduced in K-2. By the end of fifth grade, students will be able to model and discuss internal and external computing systems, as well as troubleshoot problems that may occur. Students will be able to explore and discuss real-world problems and processes related to networks and the internet. In addition, students will also be able to collect and analyze data to support inferences and models. Building on their previous understanding of algorithms and programming (coding), students will work collaboratively and independently to create and modify increasing complex programs for a variety of purposes. Students will be able explain cultural, social, and ethical impacts of computing.

By the end of 5th grade, students can:

- Describe how internal and external parts of computing devices function to form a system
- Model hardware and software information translation, transmission, and processing
- Develop, apply, and explain strategies for troubleshooting problems
- Model and explain how information is sent and received over physical or wireless paths
- Identify and implement strategies for protecting personal information
- Justify the format and location for storing different data
- Use data to highlight or propose relationships, predict outcomes, communicate an idea, or support a claim
- Collaboratively and independently create and modify (remix) programs through an iterative process
- Describe both design and debugging choices made during program development
- Explain cultural impacts of computing technologies
- Seek diverse perspectives when developing, testing, and refining digital artifacts or devices
- Work respectfully and responsibly with others in an online environment and discuss the social impact of violating intellectual property rights

WYOMING 2019 COMPUTER SCIENCE DOMAINS & STANDARDS

Computing Systems	Networks & The Internet	Data Analysis	Algorithms & Programming	Impacts of Computing
CS.D—Devices	NI.NCO—Network Communication & Organization	DA.S—Storage	AP.A—Algorithms	IC.C—Culture
CS.HS—Hardware & Software	NI.C—Cybersecurity	DA.CVT—Collection, Visualization, & Transformation	AP.V—Variables	IC.SI—Social Interactions
CS.T—Troubleshooting		DA.IM—Inference & Models	AP.C—Control	IC.SLE—Safety, Law, & Ethics
			AP.M—Modularity	
			AP.PD—Program Development	

3-5 Computer Science Practices

There are seven (7) CS Practices that are to be embedded in curriculum and instruction as the standards and benchmarks are taught and measured. The seven (7) CS Practices are listed below, and are more deeply explored on the next several pages. For each grade-band, only the CS Practices that relate are in black text and the others are grayed so the reader can still see them as a set, but will know which ones apply to that particular set of standards.

Practice 1. Fostering an Inclusive Computing Culture

Practice 2. Collaborating Around Computing

Practice 3. Recognizing and Defining Computational Problems

Practice 4. Developing and Using Abstractions

Practice 5. Creating Computational Artifacts

Practice 6. Testing and Refining Computational Artifacts

Practice 7. Communicating About Computing

DESCRIPTION OF 3-5 COMPUTER SCIENCE (CS) PRACTICES

CS Practice 1. Fostering an Inclusive Computing Culture

Overview: 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.

At all grade levels, students should recognize that the choices people make when they create artifacts are based on personal interests, experiences, and needs. Young learners should begin to differentiate their technology preferences from the technology preferences of others. Initially, students should be presented with perspectives from people with different backgrounds, ability levels, and points of view. As students progress, they should independently seek diverse perspectives throughout the design process for the purpose of improving their computational artifacts. Students who are well-versed in fostering an inclusive computing culture should be able to differentiate backgrounds and skill sets and know when to call upon others, such as to seek out knowledge about potential end users or intentionally seek input from people with diverse backgrounds.

1.2 Address the needs of diverse end users during the design process to produce artifacts with broad accessibility and usability.

At any level, students should recognize that users of technology have different needs and preferences and that not everyone chooses to use, or is able to use, the same technology products. For example, young learners, with teacher guidance, might compare a touchpad and a mouse to examine differences in usability. As students progress, they should consider the preferences of people

who might use their products. Students should be able to evaluate the accessibility of a product to a broad group of end users, such as people with various disabilities. For example, they may notice that allowing an end user to change font sizes and colors will make an interface usable for people with low vision. At the higher grades, students should become aware of professionally accepted accessibility standards and should be able to evaluate computational artifacts for accessibility. Students should also begin to identify potential bias during the design process to maximize accessibility in product design. For example, they can test an app and recommend to its designers that it respond to verbal commands to accommodate users who are blind or have physical disabilities.

1.3 Employ self- and peer-advocacy to address bias in interactions, product design, and development methods.

After students have experience identifying diverse perspectives and including unique perspectives (P1.1), they should begin to employ self-advocacy strategies, such as speaking for themselves if their needs are not met. As students progress, they should advocate for their peers when accommodations, such as an assistive-technology peripheral device, are needed for someone to use a computational artifact. Eventually, students should regularly advocate for both themselves and others.

CS Practice 2. Collaborating Around Computing

Overview: 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 Cultivate working relationships with individuals possessing diverse perspectives, skills, and personalities.

At any grade level, students should work collaboratively with others. Early on, they should learn strategies for working with team members who possess varying individual strengths. For example, with teacher support, students should begin to give each team member opportunities to contribute and to work with each other as co-learners. Eventually, students should become more sophisticated at applying strategies for mutual encouragement and support. They should express their ideas with logical reasoning and find ways to reconcile differences cooperatively. For example, when they disagree, they should ask others to explain their reasoning and work together to make respectful, mutual decisions. As they progress, students should use methods for giving all group members a chance to participate. Older students should strive to improve team efficiency and effectiveness by regularly evaluating group dynamics. They should use multiple strategies to make team dynamics more productive. For example, they can ask for the opinions of quieter team members, minimize interruptions by more talkative members, and give individuals credit for their ideas and their work.

2.2 Create team norms, expectations, and equitable workloads to increase efficiency and effectiveness.

After students have had experience cultivating working relationships within teams (P2.1), they should gain experience working in particular team roles. Early on, teachers may help guide this process by providing collaborative structures. For example, students may take turns in different roles on the project, such as note taker, facilitator, or “driver” of the computer. As students progress, they should become less dependent on the teacher assigning roles and become more adept at assigning roles within their teams. For example, they should decide together how to take turns in different roles. Eventually, students should independently organize their own teams and create common goals, expectations, and equitable workloads. They should also manage project workflow using agendas and timelines and should evaluate workflow to

identify areas for improvement.

2.3 Solicit and incorporate feedback from, and provide constructive feedback to, team members and other stakeholders.

At any level, students should ask questions of others and listen to their opinions. Early on, with teacher scaffolding, students should seek help and share ideas to achieve a particular purpose. As they progress in school, students should provide and receive feedback related to computing in constructive ways. For example, pair programming is a collaborative process that promotes giving and receiving feedback. Older students should engage in active listening by using questioning skills and should respond empathetically to others. As they progress, students should be able to receive feedback from multiple peers and should be able to differentiate opinions. Eventually, students should seek contributors from various environments. These contributors may include end users, experts, or general audiences from online forums.

2.4 Evaluate and select technological tools that can be used to collaborate on a project.

At any level, students should be able to use tools and methods for collaboration on a project. For example, in the early grades, students could collaboratively brainstorm by writing on a white-board. As students progress, they should use technological collaboration tools to manage team-work, such as knowledge-sharing tools and online project spaces. They should also begin to make decisions about which tools would be best to use and when to use them. Eventually, students should use different collaborative tools and methods to solicit input from not only team members and classmates but also others, such as participants in online forums or local communities.

CS Practice 3. Recognizing and Defining Computational Problems

Overview: 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.

At any level, students should be able to identify problems that have been solved computationally. For example, young students can discuss a technology that has changed the world, such as email or mobile phones. As they progress, they should ask clarifying questions to understand whether a problem or part of a problem can be solved using a computational approach. For example, identify real-world problems that span multiple disciplines, such as increasing bike safety with new helmet technology, and can be solved computationally.

3.2 Decompose complex real-world problems into manageable sub-problems that could integrate existing solutions or procedures.

At any grade level, students should be able to break problems down into their component parts. In the early grade levels, students should focus on breaking down simple problems. For example, in a visual programming environment, students could break down (or decompose) the steps needed to draw a shape. As students progress, they should decompose larger problems into manageable smaller problems. For example, young students may think of an animation as multiple scenes and thus create each scene independently. Students can also break down a program into subgoals: getting input from the user, processing the data, and displaying the result to the user. Eventually, as students encounter complex real-world problems that span multiple disciplines or social systems, they should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist. For example, students could create an app to solve a community problem that connects to an online database through an application programming interface (API).

3.3 Evaluate whether it is appropriate and feasible to solve a problem computationally.

After students have had some experience breaking problems down (P3.2) and

identifying subproblems that can be solved computationally (P3.1), they should begin to evaluate whether a computational solution is the most appropriate solution for a particular problem. For example, students might question whether using a computer to determine whether someone is telling the truth would be advantageous. As students progress, they should systematically evaluate the feasibility of using computational tools to solve given problems or subproblems, such as through a cost-benefit analysis. Eventually, students should include more factors in their evaluations, such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns.

CS Practice 4. Developing and Using Abstractions

Overview: 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.

Students at all grade levels should be able to recognize patterns. Young learners should be able to identify and describe repeated sequences in data or code through analogy to visual patterns or physical sequences of objects. As they progress, students should identify patterns as opportunities for abstraction, such as recognizing repeated patterns of code that could be more efficiently implemented as a loop. Eventually, students should extract common features from more complex phenomena or processes. For example, students should be able to identify common features in multiple segments of code and substitute a single segment that uses variables to account for the differences. In a procedure, the variables would take the form of parameters. When working with data, students should be able to identify important aspects and find patterns in related data sets such as crop output, fertilization methods, and climate conditions.

4.2 Evaluate existing technological functionalities and incorporate them into new designs.

At all levels, students should be able to use well-defined abstractions that hide complexity. Just as a car hides operating details, such as the mechanics of the engine, a computer program's "move" command relies on hidden details that cause an object to change location on the screen. As they progress, students should incorporate predefined functions into their designs, understanding that they do not need to know the underlying implementation details of the abstractions that they use. Eventually, students should understand the advantages of, and be comfortable using, existing functionalities (abstractions) including technological resources created by other people, such as libraries and application programming interfaces (APIs). Students should be able to evaluate existing abstractions to determine which should be incorporated into designs and how they should be incorporated. For example, students could build powerful apps by incorporating existing services, such as online databases that return geolocation coordinates of street names or food nutrition information.

4.3 Create modules and develop points of interaction that can apply to multiple situations and reduce complexity.

After students have had some experience identifying patterns (P4.1), decomposing problems (P3.2), using abstractions (P4.2), and taking advantage of existing resources (P4.2), they should begin to develop their own abstractions. As they progress, students should take advantage of opportunities to develop generalizable modules. For example, students could write more efficient programs by designing procedures that are used multiple times in the program. These procedures can be generalized by defining parameters that create different outputs for a wide range of inputs. Later on, students should be able to design systems of interacting modules, each with a well-defined role, that coordinate to accomplish a common goal. Within an object-oriented programming context, module design may include defining interactions among objects. At this stage, these modules, which combine both data and procedures, can be designed and documented for reuse in other

programs. Additionally, students can design points of interaction, such as a simple user interface, either text or graphical, that reduces the complexity of a solution and hides lower-level implementation details.

4.4 Model phenomena and processes and simulate systems to understand and evaluate potential outcomes.

Students at all grade levels should be able to represent patterns, processes, or phenomena. With guidance, young students can draw pictures to describe a simple pattern, such as sunrise and sunset, or show the stages in a process, such as brushing your teeth. They can also create an animation to model a phenomenon, such as evaporation. As they progress, students should understand that computers can model real-world phenomena, and they should use existing computer simulations to learn about real-world systems. For example, they may use a preprogrammed model to explore how parameters affect a system, such as how rapidly a disease spreads. Older students should model phenomena as systems, with rules governing the interactions within the system. Students should analyze and evaluate these models against real-world observations. For example, students might create a simple producer–consumer ecosystem model using a programming tool. Eventually, they could progress to creating more complex and realistic interactions between species, such as predation, competition, or symbiosis, and evaluate the model based on data gathered from nature.

CS Practice 5. Creating Computational Artifacts

Overview: 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.

At any grade level, students should participate in project planning and the creation of brainstorming documents. The youngest students may do so with the help of teachers. With scaffolding, students should gain greater independence and sophistication in the planning, design, and evaluation of artifacts. As learning progresses, students should systematically plan the development of a program or artifact and intentionally apply computational techniques, such as decomposition and abstraction, along with knowledge about existing approaches to artifact design. Students should be capable of reflecting on and, if necessary, modifying the plan to accommodate end goals.

- 5.2 Create a computational artifact for practical intent, personal expression, or to address a societal issue.

Students at all grade levels should develop artifacts in response to a task or a computational problem. At the earliest grade levels, students should be able to choose from a set of given commands to create simple animated stories or solve pre-existing problems. Younger students should focus on artifacts of personal importance. As they progress, student expressions should become more complex and of increasingly broader significance. Eventually, students should engage in independent, systematic use of design processes to create artifacts that solve problems with social significance by seeking input from broad audiences.

- 5.3 Modify an existing artifact to improve or customize it.

At all grade levels, students should be able to examine existing artifacts to understand what they do. As they progress, students should attempt to use existing solutions to accomplish a desired goal. For example, students could attach a programmable light sensor to a physical artifact they have created to make it respond to light. Later on, they should modify or remix parts of existing programs to develop something new or to add more advanced features and

complexity. For example, students could modify prewritten code from a single-player game to create a two-player game with slightly different rules.

CS Practice 6. Testing and Refining Computational Artifacts

Overview: 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 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.

At any grade level, students should be able to compare results to intended outcomes. Young students should verify whether given criteria and constraints have been met. As students progress, they should test computational artifacts by considering potential errors, such as what will happen if a user enters invalid input. Eventually, testing should become a deliberate process that is more iterative, systematic, and proactive. Older students should be able to anticipate errors and use that knowledge to drive development. For example, students can test their program with inputs associated with all potential scenarios.

- 6.2 Identify and fix errors using a systematic process.

At any grade level, students should be able to identify and fix errors in programs (debugging) and use strategies to solve problems with computing systems (troubleshooting). Young students could use trial and error to fix simple errors. For example, a student may try reordering the sequence of commands in a program. In a hardware context, students could try to fix a device by resetting it or checking whether it is connected to a network. As students progress, they should become more adept at debugging programs and begin to consider logic errors: cases in which a program works, but not as desired. In this way, students will examine and correct their own thinking. For

example, they might step through their program, line by line, to identify a loop that does not terminate as expected. Eventually, older students should progress to using more complex strategies for identifying and fixing errors, such as printing the value of a counter variable while a loop is running to determine how many times the loop runs.

6.3 Evaluate and refine a computational artifact multiple times to enhance its performance, reliability, usability, and accessibility.

After students have gained experience testing (P6.2), debugging, and revising (P6.1), they should begin to evaluate and refine their computational artifacts. As students progress, the process of evaluation and refinement should focus on improving performance and reliability. For example, students could observe a robot in a variety of lighting conditions to determine that a light sensor should be less sensitive. Later on, evaluation and refinement should become an iterative process that also encompasses making artifacts more usable and accessible (P1.2). For example, students can incorporate feedback from a variety of end users to help guide the size and placement of menus and buttons in a user interface.

CS Practice 7. Communicating About Computing

Overview: 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.

At any grade level, students should be able to refer to data when communicating an idea. Early on, students should, with guidance, present basic data through the use of visual representations, such as storyboards,

flowcharts, and graphs. As students progress, they should work with larger data sets and organize the data in those larger sets to make interpreting and communicating it to others easier, such as through the creation of basic data representations. Eventually, students should be able to select relevant data from large or complex data sets in support of a claim or to communicate the information in a more sophisticated manner.

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

At any grade level, students should be able to talk about choices they make while designing a computational artifact. Early on, they should use language that articulates what they are doing and identifies devices and concepts they are using with correct terminology (e.g., program, input, and debug). Younger students should identify the goals and expected outcomes of their solutions. Along the way, students should provide documentation for end users that explains their artifacts and how they function, and they should both give and receive feedback. For example, students could provide a project overview and ask for input from users. As students progress, they should incorporate clear comments in their product and document their process using text, graphics, presentations, and demonstrations.

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

All students should be able to explain the concepts of ownership and sharing. Early on, students should apply these concepts to computational ideas and creations. They should identify instances of remixing, when ideas are borrowed and iterated upon, and give proper attribution. They should also recognize the contributions of collaborators. Eventually, students should consider common licenses that place limitations or restrictions on the use of computational artifacts. For example, a downloaded image may have restrictions that prohibit modification of an image or using it for commercial purposes.



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Computing Systems

Practice(s): 7.2

By end of Grade 5

Standard: Devices	3.CS.D.01 With guidance and following directions, identify how internal and external parts of computing devices function to form a system.	4.CS.D.01 With guidance, describe how internal and external parts of computing devices function to form a system.	5.CS.D.01 Independently, describe how internal and external parts of computing devices function to form a system.
Clarification Statement: 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. Students should describe how devices and components interact using correct terminology.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV5.4.2, CV5.4.3	1d - Empowered Learner
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
RI.5.4, RI.5.7, W.5.2, W.5.4, W.5.7, W.5.8, W.5.9, SL.5.4, SL.5.5, SL.5.6, L.5.1, L.5.2, L.5.3, L.5.4, L.5.5, L.5.6			



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Computing Systems

Practice(s): 4.4

By end of Grade 5

Standard:
**Hardware &
Software**

3.CS.HS.01 Model how information flows through hardware and software to accomplish tasks.

4.CS.HS.01 Model how computer hardware and software work together as a system to accomplish tasks.

5.CS.HS.01 Model how information is translated, transmitted, and processed in order to flow through hardware and software to accomplish tasks.

Clarification Statement: In order for a person to accomplish tasks with a computer, both hardware and software are needed. At this stage, a model should only include the basic elements of a computer system, such as input, output, processor, sensors, and storage. Students could draw a model on paper or in a drawing program, program an animation to demonstrate it, or demonstrate it by acting this out in some way.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV5.4.3	1d - Empowered Learner
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH

Domain: Computing Systems
Practice(s): 6.2
By end of Grade 5

Standard: Troubleshooting	3.CS.T.01 Identify hardware and software problems that may occur during everyday use, then develop, apply, and explain strategies for solving these problems (e.g., refresh the screen, closing and reopening an application or file, unmuting or adjusting the volume on headphones).	4.CS.T.01 Identify hardware and software problems that may occur during everyday use, then develop, apply, and explain strategies for solving these problems (e.g., rebooting the device, checking the power, force shut down of an application).	5.CS.T.01 Identify hardware and software problems that may occur during everyday use, then develop, apply, and explain strategies for solving these problems.
Clarification Statement: Although computing systems may vary, common troubleshooting strategies can be used on all of them. Students should be able to identify solutions to problems such as the device not responding, no power, no network, app crashing, no sound, or password entry not working. Should errors occur at school, the goal would be that students would use various strategies, such as rebooting the device, checking for power, checking network availability, closing and reopening an app, making sure speakers are turned on or headphones are plugged in, and making sure that the caps lock key is not on, to solve these problems, when possible.			

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	3-5-ETS1-3	CV5.3.3, CV5.4.3	1d - Empowered Learner
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
SL5.6			



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Networks & the Internet

Practice(s): 4.4

By end of Grade 5

Standard: Network Communication & Organization	3.NI.NCO.01 Use observations of everyday situations to illustrate that information is sent and received over physical or wireless paths.	4.NI.NCO.01 Discuss how information is sent and received across physical or wireless path (i.e., It is broken down into smaller pieces called packets and transmitted from one location to another).	5.NI.NCO.01 Model and explain how information is broken down into smaller pieces, transmitted as packets through multiple devices over networks and the internet, and reassembled at the destination.
---	---	---	--

Clarification Statement: Information is sent and received over physical or wireless paths. It is broken down into smaller pieces called packets, which are sent independently and reassembled at the destination. Students should demonstrate their understanding of this flow of information by, for instance, drawing a model of the way packets are transmitted, programming an animation to show how packets are transmitted, or demonstrating this through an unplugged activity which has them act it out in some way.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	4-PS4-3	CV5.4.2, CV5.4.3	5c - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
SL5.6			



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Networks & the Internet

Practice(s): 3.1

By end of Grade 5

Standard: Cybersecurity	3.NI.C.01 Identify cybersecurity problems that relate to inappropriate use of computing devices and networks.	4.NI.C.01 Identify and explain cybersecurity issues related to responsible use of technology and information, and describe personal consequences of inappropriate use.	5.NI.C.01 Discuss real-world cybersecurity problems and identify and implement appropriate strategies for how personal information can be protected.
Clarification Statement: Just as we protect our personal property offline, we also need to protect our devices and the information stored on them. Information can be protected using various security measures. These measures can be physical and/or digital. Students could discuss or use a journaling or blogging activity to explain, orally or in writing, 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 anti-virus software updated to protect data and systems.			

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV5.4.3	2d - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
SL 5.1			



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Data Analysis

Practice(s): 4.2

By end of Grade 5

**Standard:
Storage**

3.DA.S.01 Demonstrate that different types of information are stored in different formats that have associated programs (e.g., documents open in a word processor) and different storage requirements.

4.DA.S.01 Choose different storage locations (physical, shared, or cloud) based on the type of file, storage requirements (e.g., file size, availability, or available memory), and sharing requirements.

5.DA.S.01 Justify the format and location for storing data based on sharing requirements and the type of information (e.g., images, videos, text).

Clarification Statement: Data may be stored locally on a computer in the classroom, on a school network, or "in the cloud." Each location affects how easily the data may be shared and how secure, or not, it is. Different types of data such as photos or documents each have their own variety of file formats which affect file size and the number and types of programs that can access that data. Students should understand and be able to explain/justify why a particular location or format is appropriate for the data they are creating or using.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV5.4.2, CV5.4.3	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
RI 5.7			



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Data Analysis

Practice(s): 7.1

By end of Grade 5

Standard: Collection, Visualization, & Transformation	3.DA.CVT.01 Independently collect and present data in various visual formats.	4.DA.CVT.01 Organize and present collected data in a variety of ways (e.g., sonification, visualization) to highlight relationships.	5.DA.CVT.01 Organize and present collected data to highlight relationships and support a claim.
<p>Clarification Statement: 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. As another example, seismographic data of an earthquake can be presented through sonification (i.e., the audible representation of data) of an earthquake's size, strength, and duration to highlight relationships and support a claim.</p>			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
3.MD.H.3	3-5-ETS1-3, 3-PS2-2, 3-LS1-1, 3-LS2-1, 3-LS3-1, 3-LS3-2, 3-LS4-1, 3-LS4-3, 3-LS4-4, 3-ESS2-1, 3-ESS3-1, 4-LS1-1, 4-ESS1-1	CV5.4.1, CV5.4.2	6a, 6c - Creative Communicator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
W 5.5	SS5.4.2		

Domain: Data Analysis
Practice(s): 7.1

By end of Grade 5

Standard: Inference & Models	3.DA.IM.01 With guidance, use data to make predictions and discuss whether there is adequate data to make reliable predictions.	4.DA.IM.01 Determine how the accuracy of conclusions is influenced by the amount of data collected.	5.DA.IM.01 Use data to highlight or propose relationships, predict outcomes, or communicate an idea.
---	--	--	---

Clarification Statement: The accuracy of data analysis is related to how realistically data is represented. Inferences or predictions based on data are less likely to be accurate if the data is not sufficient or if the data is incorrect in some way. Students should be able to refer to data when communicating an idea (e.g., 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. Another example, students could record the temperature at noon each day as a basis to show that temperatures are higher in certain months of the year. If temperatures are not recorded on non-school days or are recorded incorrectly or at different times of the day, the data would be incomplete and the ideas being communicated could be inaccurate. Students may also record the day of the week on which the data was collected, but this would have no relevance to whether temperatures are higher or lower. In order to have sufficient and accurate data on which to communicate the idea, students might want to use data provided by a governmental weather agency.).

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	3-5-ETS1-1, 3-5-ETS1-2, 3-5-ETS1-3, 3-PS2-1, 3-PS2-3, 3-PS2-4, 3-LS1-1, 3-LS2-1, 3-LS3-1, 3-LS3-2, 3-LS4-1, 3-LS4-2, 3-LS4-3, 3-LS4-4, 3-ESS2-2, 4-PS4-1, 4-PS4-2, 4-PS4-3, 4-LS1-1, 4-LS1-2, 4-ESS2-1, 4-ESS2-2, 4-ESS3-1, 4-ESS3-2	CV5.4.2, CV5.4.4	6c, 6d - Creative Communicator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
W 5.5	SS5.4.2		

Domain: Algorithms & Programming
Practice(s): 3.3, 6.3
By end of Grade 5
**Standard:
Algorithms**

3.AP.A.01 Using grade appropriate content and complexity, compare and refine multiple algorithms for the same task and determine which is the most appropriate.

4.AP.A.01 Using grade appropriate content and complexity, compare and refine multiple algorithms for the same task and determine which is the most appropriate.

5.AP.A.01 Using grade appropriate content and complexity, compare and refine multiple algorithms for the same task and determine which is the most appropriate.

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

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
3.NBT.E.2, 4.NBT.E.4, 4.NBT.E.5, 4.NBT.E.5a, 4.NBT.E.5b, 4.NBT.E.5c, 4.NBT.E.6, 4.MD.I.3, 4.MD.K.7, 5.NBT.D.5, 5.NBT.D.6, 5.NBT.D.7, 5.MD.I.5a, 5.MD.I.5b	3-5-ETS1-2, 3-5-ETS1-3, 3-PS2-1, 4-PS4-3		4a - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
		FPA4.1.A.2, FPA4.1.D.5, FPA4.1.T.1	



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Algorithms & Programming

Practice(s): 5.2

By end of Grade 5

**Standard:
Variables**

3.AP.V.01 Using grade appropriate content and complexity, create programs that use variables to store and modify data.

4.AP.V.01 Using grade appropriate content and complexity, create programs that use variables to store and modify data.

5.AP.V.01 Using grade appropriate content and complexity, create programs that use variables to store and modify data.

Clarification Statement: Variables are used to store and modify data. At this level, understanding how to use variables is sufficient. For example, students may use mathematical operations to add to the score of a game or subtract from the number of lives available in a game. The use of a variable as a countdown timer is another example.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
3.OA.D.8, 3.OA.D.8a, 4.OA.A.3, 4.OA.A.3a		CV5.4.1	4a - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



Domain: Algorithms & Programming

Practice(s): 5.2

By end of Grade 5

Standard: Control

3.AP.C.01 Using grade appropriate content and complexity, create programs that include sequences, events, loops, and conditionals, both individually and collaboratively.

4.AP.C.01 Using grade appropriate content and complexity, create programs that include sequences, events, loops, and conditionals, both individually and collaboratively.

5.AP.C.01 Using grade appropriate content and complexity, create programs that include sequences, events, loops, and conditionals, both individually and collaboratively.

Clarification Statement: 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. For example, students could write a program to explain the water cycle and when a specific component is clicked (event), the program would show information about that part of the water cycle. 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 assign a point if 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.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
3.OA.D.9, 4.OA.C.5	3-5-ETS1-2, 4-PS3-4, 4-ESS3-2	CV5.4.1	4a - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Algorithms & Programming

Practice(s): 3.2

By end of Grade 5

Standard: Modularity	3.AP.M.01 Using grade appropriate content and complexity, decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.	4.AP.M.01 Using grade appropriate content and complexity, decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.	5.AP.M.01 Using grade appropriate content and complexity, decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
---------------------------------	--	--	--

Clarification Statement: Decomposition is the act of breaking down tasks into simpler tasks. For example, students could create an animation by separating a story into different scenes. For each scene, they would select a background, position of characters, and program actions.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
3.G.K.1, 3.G.K.2, 3.OA.A.3, 4.NBT.E.4, 4.NBT.E.5, 4.NBT.E.5a, 4.NBT.E.5b, 4.NBT.E.5c, 4.MD.I.3, 4.MD.K.7, 5.G.K.3, 5.G.K.4, 5.NBT.D.5, 5.NBT.D.6, 5.NBT.D.7, 5.OA.A.1, 5.OA.A.2	3-5-ETS1-1, 3-5-ETS1-2, 3-5-ETS1-3, 3-PS2-1, 3-PS2-3, 3-PS2-4, 3-LS1-1, 3-LS3-1, 3-LS4-3, 4-PS3-4, 4-PS4-1, 4-PS4-2, 4-ESS1-1, 4-ESS2-1, 4-ESS3-1, 4-ESS3-2		5c - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: **3-5**

Domain: Algorithms & Programming

Practice(s): 5.3

By end of Grade 5

Standard: Modularity	3.AP.M.02 Using grade appropriate content and complexity, modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.	4.AP.M.02 Using grade appropriate content and complexity, modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.	5.AP.M.02 Using grade appropriate content and complexity, modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.
---------------------------------	---	---	---

Clarification Statement: Programs can be created or modified using parts from existing programs (i.e. the process of remixing). 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 from another program to make a ball bounce in a new basketball game, or modify an image created by another student.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	3-PS2-1, 3-PS2-2, 3-PS2-3, 3-PS2-4, 4-PS3-4		6b - Creative Communicator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Algorithms & Programming

Practice(s): 6.2

By end of Grade 5

Standard:
Program
Development

3.AP.PD.01 Use an iterative process to plan the development of a program.

4.AP.PD.01 Use an iterative process to plan the development of a program that includes user preferences.

5.AP.PD.01 Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences.

Clarification Statement: Planning is an important part of the iterative process of program development. Students could outline key features, time and resource constraints, and user expectations. Students could document the plan (e.g., as a storyboard, flowchart, pseudocode, or story map).

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	3-PS2-1, 3-PS2-2, 3-PS2-4		4c - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



Domain: Algorithms & Programming

Practice(s): 5.2, 7.3

By end of Grade 5

Standard: Program Development	3.AP.PD.02 Using grade appropriate content and complexity, observe intellectual property rights and give appropriate credit when creating or remixing programs.	4.AP.PD.02 Using grade appropriate content and complexity, observe intellectual property rights and give appropriate credit when creating or remixing programs.	5.AP.PD.02 Using grade appropriate content and complexity, observe intellectual property rights and give appropriate credit when creating or remixing programs.
Clarification Statement: Intellectual property rights can vary by country, but copyright laws give the creator of a work a set of rights that limits how others may use the work. Students should identify instances of remixing, when ideas are borrowed and iterated upon, and credit the original creator. Students should also consider common licenses that place limitations or restrictions on the use of computational artifacts, such as images and music downloaded from the Internet. At this stage, attribution should be written in the format required by the teacher and should always be included in any programs shared online.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	3-ESS2-1, 4-ESS3-1, 4-ESS3-2		2c - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
		FPA4.1.A.5	



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Algorithms & Programming

Practice(s): 6.1, 6.2

By end of Grade 5

Standard: Program Development	3.AP.PD.03 Using grade appropriate content and complexity, test and debug (i.e., identify and fix errors) a program or algorithm to ensure it runs as intended.	4.AP.PD.03 Using grade appropriate content and complexity, test and debug (i.e., identify and fix errors) a program or algorithm to ensure it runs as intended.	5.AP.PD.03 Using grade appropriate content and complexity, test and debug (i.e., identify and fix errors) a program or algorithm to ensure it runs as intended.
Clarification Statement: As students develop programs, they should continuously test those programs (to see that they do what was expected) and fix (debug) any errors. Students should also be able to successfully debug simple errors in programs created by others.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	3-5-ETS1-2, 3-5-ETS1-3, 3-PS2-1, 3-PS2-2, 3-5-ETS1-1, 4-PS3-4, 4-PS4-1, 4-PS4-3		4a - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
RI.5.7			



Domain: Algorithms & Programming

Practice(s): 7.2

By end of Grade 5

Standard: Program Development	3.AP.PD.04 Using grade appropriate content and complexity, describe choices made during program development using code comments, presentations, and demonstrations.	4.AP.PD.04 Using grade appropriate content and complexity, describe choices made during program development using code comments, presentations, and demonstrations.	5.AP.PD.04 Using grade appropriate content and complexity, describe choices made during program development using code comments, presentations, and demonstrations.
Clarification Statement: People communicate about their code to help others understand and use their programs. Another purpose of communicating one's design choices is to show an understanding of one's work. These explanations could manifest themselves as in-line code comments for collaborators and assessors, or as part of a summative presentation, such as a code walk-through or coding journal.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	3-5-ETS1-1, 3-5-ETS1-2, 3-5-ETS1-3, 3-PS2-1, 3-PS2-4, 4-PS3-4, 4-PS4-1, 4-PS4-3	CV5.4.2, CV5.4.3	3d - Knowledge Constructor
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
RI.5.3, W.5.6		FPA4.1.A.6	



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Algorithms & Programming

Practice(s): 2.2

By end of Grade 5

Standard: Program Development	3.AP.PD.05 Using grade appropriate content and complexity, with teacher guidance, perform varying roles when collaborating with peers during the design, implementation, and review stages of program development.	4.AP.PD.05 Using grade appropriate content and complexity, with teacher guidance, perform varying roles when collaborating with peers during the design, implementation, and review stages of program development.	5.AP.PD.05 Using grade appropriate content and complexity, with teacher guidance, perform varying roles when collaborating with peers during the design, implementation, and review stages of program development.
Clarification Statement: Collaborative computing is the process of performing a computational task by working in pairs or on teams. Because it involves asking for the contributions and feedback of others, effective collaboration can lead to better outcomes than working independently. Students should take turns in different roles during program development, such as note taker, facilitator, program tester, or “driver” of the computer.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	3-5-ETS1-2, 3-5-ETS1-3, 3-PS2-1, 3-PS2-2, 3-PS2-3, 3-LS1-1		7a - Global Collaborator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
W.5.6		FPA4.1.A.4, FPA4.1.A.6, FPA4.1.T.4	



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Impacts of Computing

Practice(s): 3.1

By end of Grade 5

Standard: Culture

3.IC.C.01 Identify computing technologies that have changed the world and express how those technologies influence and are influenced by cultural practices.

4.IC.C.01 Give examples of computing technologies that have changed the world and express how those technologies influence and are influenced by cultural practices.

5.IC.C.01 Give examples and explain how computing technologies have changed the world and express how those technologies influence and are influenced by cultural practices.

Clarification Statement: New computing technology is created and existing technologies are modified for many reasons (e.g., increasing their benefits, decreasing their risks, and meeting societal needs). With guidance, students could 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, artificial intelligence, or how social media has influenced social and political changes.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	3-5-ETS1-1	CV5.4.2	7a - Global Collaborator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
RI.5.9, W.5.1	SS5.3.3, SS5.4.2	FPA4.3.A.3, FPA4.3.M.3	



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Impacts of Computing

Practice(s): 1.2

By end of Grade 5

**Standard:
Culture**

3.IC.C.02 Identify possible problems and propose how computing devices could have built-in features for increasing accessibility to all users.

4.IC.C.02 Brainstorm problems and ways to improve computing devices to increase accessibility to all users.

5.IC.C.02 Develop, test, and refine digital artifacts or devices to improve accessibility and usability for diverse end users.

Clarification Statement: The development and modification of computing technologies are driven by people's needs and wants and can affect groups differently. Anticipating the needs and wants of diverse end users requires students to purposefully consider potential perspectives of users with different backgrounds, ability levels, points of view, and disabilities. For example, students may consider using both speech and text when they wish to convey information in a game. They could also vary the types of programs they create, knowing that not everyone shares their own tastes or (dis)abilities.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
			4c - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



Domain: Impacts of Computing

Practice(s): 1.1

By end of Grade 5

Standard: Social Interactions	3.IC.SI.01 Identify how computational products may be, or have been, improved to incorporate diverse perspectives.	4.IC.SI.01 As a team or individually, consider other perspectives on improving a computational product.	5.IC.SI.01 Seek diverse perspectives for the purpose of improving computational artifacts.
--	---	--	---

Clarification Statement: Computing provides the possibility for collaboration and sharing of ideas and allows the benefit of diverse perspectives. For example, students could seek feedback from other groups in their class or students at another grade level. With guidance, students could use video conferencing tools or other online collaborative spaces (e.g., blogs, wikis, forums, or website comments) to gather feedback from individuals and groups about programming projects.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
			7a - Global Collaborator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			PE 5.3.2



2019 Wyoming Computer Science Standards

Grade Band: 3-5

Domain: Impacts of Computing

Practice(s): 2.1

By end of Grade 5

Standard: Social Interactions	3.IC.SI.02 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.	4.IC.SI.02 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.	5.IC.SI.02 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.
Clarification Statement: Examples of "online communities" can be small and simple, such as using a shared hard drive or thumb drive within the classroom. They can also extend to sharing folders in cloud-based storage, writing for a school-wide blog, or collaborating with another classroom across the country or around the world using video conferencing. Examples of inappropriate behavior might include sharing another person's private data, providing inappropriate feedback on another person's project, or posting content under another person's name or account.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV5.2.3, CV5.2.4, CV5.5.3, CV5.5.4	2b - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
		FPA4.1.A.5, FPA4.4.A.4, FPA4.4.M.1, FPA4.4.T.2	PE 5.3.2 HE4.4.9



Domain: Impacts of Computing

Practice(s): 7.3

By end of Grade 5

Standard: Safety, Law, & Ethics	3.IC.SLE.01 Identify types of digital data that may have intellectual property rights that prevent copying or require attribution.	4.IC.SLE.01 Recognize and appropriately use public domain and/or creative commons media and discuss the social impact of violating intellectual property rights.	5.IC.SLE.01 Recognize and appropriately use public domain and creative commons media and discuss the social impact of violating intellectual property rights.
Clarification Statement: 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 the disregard of copyrights. Students should consider the licenses on computational artifacts that they wish to use. For example, the license on a downloaded image or audio file may have restrictions that prohibit modification, require attribution, or prohibit use entirely.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV5.2.4, CV5.5.3, CV5.5.4	2c - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH

Performance Level Descriptors (PLDs)

Grade Band: 3-5

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Devices: 5.CS.D.01 Independently, describe how internal and external parts of computing devices function to form a system.	provides little to no evidence in addressing the expectation(s).	with guidance, describes with some errors how internal and external parts of computing devices function to form a system.	independently describes with few to no errors how internal and external parts of computing devices function to form a system.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., demonstrates on different types of devices).
Hardware & Software: 5.CS.HS.01 Model how information is translated, transmitted, and processed in order to flow through hardware and software to accomplish tasks.	provides little to no evidence in addressing the expectation(s).	partially models how information is translated, transmitted, and processed in order to flow through hardware and software to accomplish tasks.	accurately models how information is translated, transmitted, and processed in order to flow through hardware and software to accomplish tasks.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., compare and contrast different devices).
Troubleshooting: 5.CS.T.01 Identify hardware and software problems that may occur during everyday use, then develop, apply, and explain strategies for solving these problems.	provides little to no evidence in addressing the expectation(s).	partially: - identifies hardware and software problems that may occur during everyday use. - attempts to solve identified problems, when applicable.	accurately: - identifies hardware and software problems that may occur during everyday use. - develops, applies, and explains strategies for solving identified problems, when applicable.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., develops a troubleshooting guide, helps others with troubleshooting issues efficiently, suggests preventative measures).

Performance Level Descriptors (PLDs)

Grade Band: 3-5

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Network Communication Organization: 5.NI.NCO.01 Model and explain how information is broken down into smaller pieces, transmitted as packets through multiple devices over networks and the internet, and reassembled at the destination.	provides little to no evidence in addressing the expectation(s).	partially models and explains how information is: - broken down into smaller pieces, and - transmitted as packets through multiple devices over networks and the internet, and/or - reassembled at the destination.	accurately models and explains how information is: - broken down into smaller pieces. - transmitted as packets through multiple devices over networks and the internet. - reassembled at the destination.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., compares and contrasts different connection types).
Cybersecurity: 5.NI.C.01 Discuss real-world cybersecurity problems and identify and implement appropriate strategies for how personal information can be protected.	provides little to no evidence in addressing the expectation(s).	- generally discusses real-world cybersecurity problems, and/or - identifies appropriate strategies for how personal information can be protected.	- discusses with specificity real-world cybersecurity problems. - discusses personal consequences of inappropriate use. - identifies and implements appropriate strategies for how personal information can be protected.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., compares and contrasts a variety of approaches to authentication, evaluates current practices).

Performance Level Descriptors (PLDs)

Grade Band: 3-5

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Storage: 5.DA.S.01 Justify the format and location for storing data based on sharing requirements and the type of information (e.g., images, videos, text).	provides little to no evidence in addressing the expectation(s).	describes the format, location, sharing requirements, or the type of information when storing data.	justifies the format and location for storing data based on sharing requirements and the type of information.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., determines the best file type for a given purpose, suggests strategies to solve a problem, creates a document in a variety of formats, converts files).
Collection, Visualization, & Transformation: 5.DA.CVT.01 Organize and present collected data to highlight relationships and support a claim.	provides little to no evidence in addressing the expectation(s).	organizes and presents collected data.	organizes and presents collected data to: - highlight comparisons. - highlight relationships. - to support a claim.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., helps others organize collected data, suggests improvements on how to organize collected data to a specific audience).
Inference & Models: 5.DA.IM.01 Use data to highlight or propose relationships, predict outcomes, or communicate an idea.	provides little to no evidence in addressing the expectation(s).	with guidance, uses data to: - highlight relationships, and/or - communicate an idea.	independently uses data to: - highlight or propose relationships, and/or - predict outcomes, and/or - communicate an idea.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., proposes alternative models, proposes additional factors that could affect a relationship).

Performance Level Descriptors (PLDs)

Grade Band: 3-5

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Algorithms: 5.AP.A.01 Using grade appropriate content and complexity, compare and refine multiple algorithms for the same task and determine which is the most appropriate.	provides little to no evidence in addressing the expectation(s).	compares simple algorithms for the same task.	<ul style="list-style-type: none"> - compares and refines multiple algorithms for the same task. - determines which algorithm is the most appropriate for the same task. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., develop alternative algorithms).
Variables: 5.AP.V.01 Using grade appropriate content and complexity, create programs that use variables to store and modify data.	provides little to no evidence in addressing the expectation(s).	modifies programs that use variables to: <ul style="list-style-type: none"> - store data. - modify data. 	creates programs that use variables to: <ul style="list-style-type: none"> - store data. - modify data. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., uses a variety of variable types).
Control: 5.AP.C.01 Using grade appropriate content and complexity, create programs that include sequences, events, loops, and conditionals, both individually and collaboratively.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - independently, create programs that include sequences and events. - collaboratively, create programs that include sequences and events. 	<ul style="list-style-type: none"> - independently, create programs that include combinations of sequences, events, loops, and conditionals. - collaboratively, create programs that include combinations of sequences, events, loops, and conditionals. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., incorporating nested loops and complex conditionals).

Performance Level Descriptors (PLDs)

Grade Band: 3-5

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
<p>Modularity: 5.AP.M.01 Using grade appropriate content and complexity, decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.</p> <p>Modularity: 5.AP.M.02 Using grade appropriate content and complexity, modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.</p>	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - decomposes (breaks down) problems into smaller, manageable subproblems to facilitate the program development process, <p>and/or</p> <ul style="list-style-type: none"> - modifies, remixes, or incorporates portions of an existing program into one's own work. 	<ul style="list-style-type: none"> - decomposes (breaks down) problems into smaller, manageable subproblems to facilitate the program development process. - modifies, remixes, or incorporates portions of an existing program into one's own work to develop something new or add more advanced features. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., helps others modify code, incorporates portions of multiple programs).

Performance Level Descriptors (PLDs)

Grade Band: 3-5

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
<p>Program Development: 5.AP.PD.01 Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences.</p> <p>Program Development: 5.AP.PD.02 Using grade appropriate content and complexity, observe intellectual property rights and give appropriate credit when creating or remixing programs.</p> <p>Program Development: 5.AP.PD.03 Using grade appropriate content and complexity, test and debug (i.e., identify and fix errors) a program or algorithm to ensure it runs as intended.</p> <p>Program Development: 5.AP.PD.04 Using grade appropriate content and complexity, describe choices made during program development using code comments, presentations, and demonstrations.</p> <p>Program Development: 5.AP.PD.05 Using grade appropriate content and complexity, with teacher guidance, perform varying roles when collaborating with peers during the design, implementation, and review stages of program development.</p>	<p>provides little to no evidence in addressing the expectation(s).</p>	<ul style="list-style-type: none"> - observes intellectual property rights and gives appropriate credit when creating or remixing programs, and - uses an iterative process to plan the development of a program by including other perspectives and considers user preferences, and/or - tests and debugs (identify and fix errors) a program or algorithm to ensure it runs as intended, and/or - describes choices made during program development using code comments, presentations, and demonstrations, and/or - with teacher guidance, performs varying roles when collaborating with peers during the design, implementation, and review stages of program development. 	<ul style="list-style-type: none"> - observes intellectual property rights and gives appropriate credit when creating or remixing programs. - uses an iterative process to plan the development of a program by including other perspectives and considers user preferences. - tests and debugs (identify and fix errors) a program or algorithm to ensure it runs as intended. - describes choices made during program development using code comments, presentations, and demonstrations. - with teacher guidance, performs varying roles when collaborating with peers during the design, implementation, and review stages of program development. 	<p>demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard. By way of examples,</p> <ul style="list-style-type: none"> - justifies their own copyright on their work; -explains the different types of copyrights and the process of getting permission; - provides guidance to other students when testing and debugging a program or algorithm; - proposes alternatives and justifies why they went with their current code.

Performance Level Descriptors (PLDs)

Grade Band: 3-5

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
<p>Culture: 5.IC.C.01 Give examples and explain how computing technologies have changed the world and express how those technologies influence and are influenced by cultural practices.</p> <p>Culture: 5.IC.C.02 Develop, test, and refine digital artifacts or devices to improve accessibility and usability for diverse end users.</p>	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - gives examples of how computing technologies have changed the world, <p>and/or</p> <ul style="list-style-type: none"> - expresses how technologies interact with cultural practices, <p>and/or</p> <ul style="list-style-type: none"> - tests digital artifacts or devices for accessibility and usability for diverse end users. 	<ul style="list-style-type: none"> - gives examples and explains how computing technologies have changed the world. - expresses how technologies influence and are influenced by cultural practices. - develops, tests, and refines digital artifacts or devices to improve accessibility and usability for diverse end users. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., makes and justifies predictions based on historical patterns, incorporates multiple forms of accessibility in one artifact).
<p>Social Interactions: 5.IC.SI.01 Seek diverse perspectives for the purpose of improving computational artifacts.</p> <p>Social Interactions: 5.IC.SI.02 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.</p>	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - practices grade-level appropriate behavior and responsibilities while participating in an online community. - identifies and reports inappropriate behavior, when applicable. 	<ul style="list-style-type: none"> - seeks diverse perspectives for the purpose of improving computational artifacts. - practices grade-level appropriate behavior and responsibilities while participating in an online community. - identifies and reports inappropriate behavior, when applicable. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., creates resources that models or explains to peers how to participate in online communities or independently uses video conferencing tools or other online collaborative spaces, such as blogs, wikis, forums, or website comments, to gather feedback from individuals and groups).

Performance Level Descriptors (PLDs)

Grade Band: 3-5

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Safety, Law, & Ethics: 5.IC.SLE.01 Recognize and appropriately use public domain and creative commons media and discuss the social impact of violating intellectual property rights.	provides little to no evidence in addressing the expectation(s).	identifies types of digital data that may have intellectual property rights that prevent copying or require attribution.	<ul style="list-style-type: none">- recognizes and appropriately uses public domain and creative commons media.- discusses the social impact of violating intellectual property rights.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., explain the process of contributing to a public domain or creative commons media, create and use a custom intellectual property rights system used by members of the class).

Computer Science | 6-8 Introduction

Throughout grades 6-8, students continue to develop their understanding of algorithms and programming (coding). Students work collaboratively and independently to create and modify increasingly complex programs for a variety of purposes introduced in grades 3-5.

By the end of 8th grade, students can:

- Systematically identify, recommend, resolve, and document increasingly complex software and hardware problems with computing devices and their components
- Model the role of protocols in transmitting data across networks and the internet
- Critique physical and digital procedures that could be implemented to protect electronic data/information
- Use and refine computational tools to transform collected data in order to make it more useful and reliable
- Create flowcharts and pseudocode to design algorithms to solve complex problems
- Create clearly named variables that represent different data types and perform operations on their values
- Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals
- Decompose problems into parts to facilitate the design, implementation, and review of programs
- Create procedures with parameters to organize code and make it easier to reuse
- Seek and incorporate feedback from team members and users to refine a solution to a problem
- Describe impacts associated with computing technologies that affect people's everyday activities and career options along with issues of bias and accessibility in the design of technologies
- Practice grade-level appropriate behavior and responsibilities while participating in an online community, including identifying and reporting inappropriate behavior
- Describe tradeoffs between allowing information to be public and keeping information private and secure
- Discuss the legal, social, and ethical impacts associated with software development and use, including both positive and malicious intent

WY 2019 COMPUTER SCIENCE DOMAINS & STANDARDS

Computing Systems	Networks & The Internet	Data Analysis	Algorithms & Programming	Impacts of Computing
CS.D—Devices CS.HS—Hardware & Software CS.T—Troubleshooting	NI.NCO—Network Communication & Organization NI.C—Cybersecurity	DA.S—Storage DA.CVT—Collection, Visualization, & Transformation DA.IM—Inference & Models	AP.A—Algorithms AP.V—Variables AP.C—Control AP.M—Modularity AP.PD—Program Development	IC.C—Culture IC.SI—Social Interactions IC.SLE—Safety, Law, & Ethics

6-8 Computer Science Practices

There are seven (7) CS Practices that are to be embedded in curriculum and instruction as the standards and benchmarks are taught and measured. The seven (7) CS Practices are listed below, and are more deeply explored on the next several pages. For each grade-band, only the CS Practices that relate are in black text and the others are grayed so the reader can still see them as a set, but will know which ones apply to that particular set of standards.

Practice 1. Fostering an Inclusive Computing Culture

Practice 2. Collaborating Around Computing

Practice 3. Recognizing and Defining Computational Problems

Practice 4. Developing and Using Abstractions

Practice 5. Creating Computational Artifacts

Practice 6. Testing and Refining Computational Artifacts

Practice 7. Communicating About Computing

DESCRIPTION OF 6-8 COMPUTER SCIENCE (CS) PRACTICES

CS Practice 1. Fostering an Inclusive Computing Culture

Overview: 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.

At all grade levels, students should recognize that the choices people make when they create artifacts are based on personal interests, experiences, and needs. Young learners should begin to differentiate their technology preferences from the technology preferences of others. Initially, students should be presented with perspectives from people with different backgrounds, ability levels, and points of view. As students progress, they should independently seek diverse perspectives throughout the design process for the purpose of improving their computational artifacts. Students who are well-versed in fostering an inclusive computing culture should be able to differentiate backgrounds and skill sets and know when to call upon others, such as to seek out knowledge about potential end users or intentionally seek input from people with diverse backgrounds.

1.2 Address the needs of diverse end users during the design process to produce artifacts with broad accessibility and usability.

At any level, students should recognize that users of technology have different needs and preferences and that not everyone chooses to use, or is able to use, the same technology products. For example, young learners, with teacher guidance, might compare a touchpad and a mouse to examine differences in usability. As students progress, they should consider the preferences of people

who might use their products. Students should be able to evaluate the accessibility of a product to a broad group of end users, such as people with various disabilities. For example, they may notice that allowing an end user to change font sizes and colors will make an interface usable for people with low vision. At the higher grades, students should become aware of professionally accepted accessibility standards and should be able to evaluate computational artifacts for accessibility. Students should also begin to identify potential bias during the design process to maximize accessibility in product design. For example, they can test an app and recommend to its designers that it respond to verbal commands to accommodate users who are blind or have physical disabilities.

1.3 Employ self- and peer-advocacy to address bias in interactions, product design, and development methods.

After students have experience identifying diverse perspectives and including unique perspectives (P1.1), they should begin to employ self-advocacy strategies, such as speaking for themselves if their needs are not met. As students progress, they should advocate for their peers when accommodations, such as an assistive-technology peripheral device, are needed for someone to use a computational artifact. Eventually, students should regularly advocate for both themselves and others.

CS Practice 2. Collaborating Around Computing

Overview: 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 Cultivate working relationships with individuals possessing diverse perspectives, skills, and personalities.

At any grade level, students should work collaboratively with others. Early on, they should learn strategies for working with team members who possess varying individual strengths. For example, with teacher support, students should begin to give each team member opportunities to contribute and to work with each other as co-learners. Eventually, students should become more sophisticated at applying strategies for mutual encouragement and support. They should express their ideas with logical reasoning and find ways to reconcile differences cooperatively. For example, when they disagree, they should ask others to explain their reasoning and work together to make respectful, mutual decisions. As they progress, students should use methods for giving all group members a chance to participate. Older students should strive to improve team efficiency and effectiveness by regularly evaluating group dynamics. They should use multiple strategies to make team dynamics more productive. For example, they can ask for the opinions of quieter team members, minimize interruptions by more talkative members, and give individuals credit for their ideas and their work.

2.2 Create team norms, expectations, and equitable workloads to increase efficiency and effectiveness.

After students have had experience cultivating working relationships within teams (P2.1), they should gain experience working in particular team roles. Early on, teachers may help guide this process by providing collaborative structures. For example, students may take turns in different roles on the project, such as note taker, facilitator, or “driver” of the computer. As students progress, they should become less dependent on the teacher assigning roles and become more adept at assigning roles within their teams. For example, they should decide together how to take turns in different roles. Eventually, students should independently organize their own teams and create common goals, expectations, and equitable workloads. They should also manage project workflow using agendas and timelines and should evaluate workflow to

identify areas for improvement.

2.3 Solicit and incorporate feedback from, and provide constructive feedback to, team members and other stakeholders.

At any level, students should ask questions of others and listen to their opinions. Early on, with teacher scaffolding, students should seek help and share ideas to achieve a particular purpose. As they progress in school, students should provide and receive feedback related to computing in constructive ways. For example, pair programming is a collaborative process that promotes giving and receiving feedback. Older students should engage in active listening by using questioning skills and should respond empathetically to others. As they progress, students should be able to receive feedback from multiple peers and should be able to differentiate opinions. Eventually, students should seek contributors from various environments. These contributors may include end users, experts, or general audiences from online forums.

2.4 Evaluate and select technological tools that can be used to collaborate on a project.

At any level, students should be able to use tools and methods for collaboration on a project. For example, in the early grades, students could collaboratively brainstorm by writing on a white-board. As students progress, they should use technological collaboration tools to manage team-work, such as knowledge-sharing tools and online project spaces. They should also begin to make decisions about which tools would be best to use and when to use them. Eventually, students should use different collaborative tools and methods to solicit input from not only team members and classmates but also others, such as participants in online forums or local communities.

CS Practice 3. Recognizing and Defining Computational Problems

Overview: 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.

At any level, students should be able to identify problems that have been solved computationally. For example, young students can discuss a technology that has changed the world, such as email or mobile phones. As they progress, they should ask clarifying questions to understand whether a problem or part of a problem can be solved using a computational approach. For example, identify real-world problems that span multiple disciplines, such as increasing bike safety with new helmet technology, and can be solved computationally.

3.2 Decompose complex real-world problems into manageable sub-problems that could integrate existing solutions or procedures.

At any grade level, students should be able to break problems down into their component parts. In the early grade levels, students should focus on breaking down simple problems. For example, in a visual programming environment, students could break down (or decompose) the steps needed to draw a shape. As students progress, they should decompose larger problems into manageable smaller problems. For example, young students may think of an animation as multiple scenes and thus create each scene independently. Students can also break down a program into subgoals: getting input from the user, processing the data, and displaying the result to the user. Eventually, as students encounter complex real-world problems that span multiple disciplines or social systems, they should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist. For example, students could create an app to solve a community problem that connects to an online database through an application programming interface (API).

3.3 Evaluate whether it is appropriate and feasible to solve a problem computationally.

After students have had some experience breaking problems down (P3.2) and

identifying subproblems that can be solved computationally (P3.1), they should begin to evaluate whether a computational solution is the most appropriate solution for a particular problem. For example, students might question whether using a computer to determine whether someone is telling the truth would be advantageous. As students progress, they should systematically evaluate the feasibility of using computational tools to solve given problems or subproblems, such as through a cost-benefit analysis. Eventually, students should include more factors in their evaluations, such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns.

CS Practice 4. Developing and Using Abstractions

Overview: 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.

Students at all grade levels should be able to recognize patterns. Young learners should be able to identify and describe repeated sequences in data or code through analogy to visual patterns or physical sequences of objects. As they progress, students should identify patterns as opportunities for abstraction, such as recognizing repeated patterns of code that could be more efficiently implemented as a loop. Eventually, students should extract common features from more complex phenomena or processes. For example, students should be able to identify common features in multiple segments of code and substitute a single segment that uses variables to account for the differences. In a procedure, the variables would take the form of parameters. When working with data, students should be able to identify important aspects and find patterns in related data sets such as crop output, fertilization methods, and climate conditions.

4.2 Evaluate existing technological functionalities and incorporate them into new designs.

At all levels, students should be able to use well-defined abstractions that hide complexity. Just as a car hides operating details, such as the mechanics of the engine, a computer program's "move" command relies on hidden details that cause an object to change location on the screen. As they progress, students should incorporate predefined functions into their designs, understanding that they do not need to know the underlying implementation details of the abstractions that they use. Eventually, students should understand the advantages of, and be comfortable using, existing functionalities (abstractions) including technological resources created by other people, such as libraries and application programming interfaces (APIs). Students should be able to evaluate existing abstractions to determine which should be incorporated into designs and how they should be incorporated. For example, students could build powerful apps by incorporating existing services, such as online databases that return geolocation coordinates of street names or food nutrition information.

4.3 Create modules and develop points of interaction that can apply to multiple situations and reduce complexity.

After students have had some experience identifying patterns (P4.1), decomposing problems (P3.2), using abstractions (P4.2), and taking advantage of existing resources (P4.2), they should begin to develop their own abstractions. As they progress, students should take advantage of opportunities to develop generalizable modules. For example, students could write more efficient programs by designing procedures that are used multiple times in the program. These procedures can be generalized by defining parameters that create different outputs for a wide range of inputs. Later on, students should be able to design systems of interacting modules, each with a well-defined role, that coordinate to accomplish a common goal. Within an object-oriented programming context, module design may include defining interactions among objects. At this stage, these modules, which combine both data and procedures, can be designed and documented for reuse in other

programs. Additionally, students can design points of interaction, such as a simple user interface, either text or graphical, that reduces the complexity of a solution and hides lower-level implementation details.

4.4 Model phenomena and processes and simulate systems to understand and evaluate potential outcomes.

Students at all grade levels should be able to represent patterns, processes, or phenomena. With guidance, young students can draw pictures to describe a simple pattern, such as sunrise and sunset, or show the stages in a process, such as brushing your teeth. They can also create an animation to model a phenomenon, such as evaporation. As they progress, students should understand that computers can model real-world phenomena, and they should use existing computer simulations to learn about real-world systems. For example, they may use a preprogrammed model to explore how parameters affect a system, such as how rapidly a disease spreads. Older students should model phenomena as systems, with rules governing the interactions within the system. Students should analyze and evaluate these models against real-world observations. For example, students might create a simple producer–consumer ecosystem model using a programming tool. Eventually, they could progress to creating more complex and realistic interactions between species, such as predation, competition, or symbiosis, and evaluate the model based on data gathered from nature.

CS Practice 5. Creating Computational Artifacts

Overview: 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.

At any grade level, students should participate in project planning and the creation of brainstorming documents. The youngest students may do so with the help of teachers. With scaffolding, students should gain greater independence and sophistication in the planning, design, and evaluation of artifacts. As learning progresses, students should systematically plan the development of a program or artifact and intentionally apply computational techniques, such as decomposition and abstraction, along with knowledge about existing approaches to artifact design. Students should be capable of reflecting on and, if necessary, modifying the plan to accommodate end goals.

5.2 Create a computational artifact for practical intent, personal expression, or to address a societal issue.

Students at all grade levels should develop artifacts in response to a task or a computational problem. At the earliest grade levels, students should be able to choose from a set of given commands to create simple animated stories or solve pre-existing problems. Younger students should focus on artifacts of personal importance. As they progress, student expressions should become more complex and of increasingly broader significance. Eventually, students should engage in independent, systematic use of design processes to create artifacts that solve problems with social significance by seeking input from broad audiences.

5.3 Modify an existing artifact to improve or customize it.

At all grade levels, students should be able to examine existing artifacts to understand what they do. As they progress, students should attempt to use existing solutions to accomplish a desired goal. For example, students could attach a programmable light sensor to a physical artifact they have created to make it respond to light. Later on, they should modify or remix parts of existing programs to develop something new or to add more advanced features and

complexity. For example, students could modify prewritten code from a single-player game to create a two-player game with slightly different rules.

CS Practice 6. Testing and Refining Computational Artifacts

Overview: 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 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.

At any grade level, students should be able to compare results to intended outcomes. Young students should verify whether given criteria and constraints have been met. As students progress, they should test computational artifacts by considering potential errors, such as what will happen if a user enters invalid input. Eventually, testing should become a deliberate process that is more iterative, systematic, and proactive. Older students should be able to anticipate errors and use that knowledge to drive development. For example, students can test their program with inputs associated with all potential scenarios.

6.2 Identify and fix errors using a systematic process.

At any grade level, students should be able to identify and fix errors in programs (debugging) and use strategies to solve problems with computing systems (troubleshooting). Young students could use trial and error to fix simple errors. For example, a student may try reordering the sequence of commands in a program. In a hardware context, students could try to fix a device by resetting it or checking whether it is connected to a network. As students progress, they should become more adept at debugging programs and begin to consider logic errors: cases in which a program works, but not as desired. In this way, students will examine and correct their own thinking. For

example, they might step through their program, line by line, to identify a loop that does not terminate as expected. Eventually, older students should progress to using more complex strategies for identifying and fixing errors, such as printing the value of a counter variable while a loop is running to determine how many times the loop runs.

6.3 Evaluate and refine a computational artifact multiple times to enhance its performance, reliability, usability, and accessibility.

After students have gained experience testing (P6.2), debugging, and revising (P6.1), they should begin to evaluate and refine their computational artifacts. As students progress, the process of evaluation and refinement should focus on improving performance and reliability. For example, students could observe a robot in a variety of lighting conditions to determine that a light sensor should be less sensitive. Later on, evaluation and refinement should become an iterative process that also encompasses making artifacts more usable and accessible (P1.2). For example, students can incorporate feedback from a variety of end users to help guide the size and placement of menus and buttons in a user interface.

CS Practice 7. Communicating About Computing

Overview: 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.

At any grade level, students should be able to refer to data when communicating an idea. Early on, students should, with guidance, present basic data through the use of visual representations, such as storyboards,

flowcharts, and graphs. As students progress, they should work with larger data sets and organize the data in those larger sets to make interpreting and communicating it to others easier, such as through the creation of basic data representations. Eventually, students should be able to select relevant data from large or complex data sets in support of a claim or to communicate the information in a more sophisticated manner.

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

At any grade level, students should be able to talk about choices they make while designing a computational artifact. Early on, they should use language that articulates what they are doing and identifies devices and concepts they are using with correct terminology (e.g., program, input, and debug). Younger students should identify the goals and expected outcomes of their solutions. Along the way, students should provide documentation for end users that explains their artifacts and how they function, and they should both give and receive feedback. For example, students could provide a project overview and ask for input from users. As students progress, they should incorporate clear comments in their product and document their process using text, graphics, presentations, and demonstrations.

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

All students should be able to explain the concepts of ownership and sharing. Early on, students should apply these concepts to computational ideas and creations. They should identify instances of remixing, when ideas are borrowed and iterated upon, and give proper attribution. They should also recognize the contributions of collaborators. Eventually, students should consider common licenses that place limitations or restrictions on the use of computational artifacts. For example, a downloaded image may have restrictions that prohibit modification of an image or using it for commercial purposes.



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Computing Systems

Practice(s): 3.3

By end of Grade 8

**Standard:
Devices**

6.CS.D.01 Recommend improvements to the design of computing devices based on analysis of personal interaction with the device.

7.CS.D.01 Recommend improvements to the design of computing devices based on analysis of how peers interact with the device.

8.CS.D.01 Recommend improvements to the design of computing devices based on an analysis of how a variety of users interact with the device.

Clarification Statement: The study of human–computer interaction (HCI) can improve the design of devices, including both hardware and software. Students should make recommendations for existing devices (e.g., a laptop, phone, or tablet) or design their own components or interface (e.g., create their own controllers). Teachers can guide students to consider usability through several lenses, including accessibility, ergonomics, and ease of use. For example, assistive devices provide capabilities such as scanning written information and converting it to speech.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	MS-ETS1-4	CV8.2.1, CV8.5.3	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
SL 8.1			



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Computing Systems

Practice(s): 5.1

By end of Grade 8

**Standard:
Hardware &
Software**

6.CS.HS.01 Identify ways that hardware and software are combined to collect and exchange data.

7.CS.HS.01 Recommend improvements to software and hardware combinations used to collect and exchange data.

8.CS.HS.01 Design and refine a project that combines hardware and software components to collect and exchange data.

Clarification Statement: Collecting and exchanging data involves input, output, storage, and processing. When possible, students should select the hardware and software components for their project designs by considering factors such as functionality, cost, size, speed, accessibility, and aesthetics. For example, components for a mobile app could include accelerometer, GPS, and speech recognition. The choice of a device that connects wirelessly through a Bluetooth connection versus a physical USB connection involves a tradeoff between mobility and the need for an additional power source for the wireless device.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	MS-ETS1-3	CV8.4.4, CV8.5.3, CV8.5.4	4c - Innovative Designer 5b - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			PE 8.3.3



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Computing Systems

Practice(s): 6.2

By end of Grade 8

Standard: Troubleshooting	6.CS.T.01 Identify and determine potential solutions for increasingly complex software and hardware problems with computing devices and their components.	7.CS.T.01 Identify and resolve increasingly complex software and hardware problems with computing devices and their components.	8.CS.T.01 Systematically identify, resolve, and document increasingly complex software and hardware problems with computing devices and their components.
--------------------------------------	--	--	--

Clarification Statement: Since a computing device may interact with interconnected devices within a system, problems may not be due to the specific computing device itself but to devices connected to it. Just as pilots use checklists to troubleshoot problems with aircraft systems, students should use a similar, structured process to troubleshoot problems with computing systems and ensure that potential solutions are not overlooked. Examples of troubleshooting strategies include following a troubleshooting flow diagram, making changes to software to see if hardware will work, checking connections and settings, and swapping in working components.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	MS-ETS1-1, MS-ETS1-4, MS-ETS2-1	CV8.2.1, CV8.3.1, CV8.4.3, CV8.5.3, CV8.5.4	3d - Knowledge Constructor
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			PE 8.3.3



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Networks & the Internet

Practice(s): 4.4

By end of Grade 8

Standard: Network Communication & Organization	6.NI.NCO.01 Model the role of protocols in transmitting data across networks and the internet (e.g., model a simple protocol for transferring information using packets).	7.NI.NCO.01 Model the role of protocols in transmitting data across networks and the internet (e.g., explain how a system responds when a packet is lost and the effect it has on the transferred information).	8.NI.NCO.01 Model the role of protocols in transmitting data across networks and the internet (e.g., explain protocols and their importance to data transmission; model how packets are broken down into smaller pieces and how they are delivered).

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

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	MS-PS4-3	CV8.2.1, CV8.4.3, CV8.5.3	5c - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Networks & the Internet

Practice(s): 7.2

By end of Grade 8

Standard: Cybersecurity	6.NI.C.01 Identify existing cybersecurity concerns with the internet.	7.NI.C.01 Explain how to protect electronic information, both physical and digital. Identify cybersecurity concerns and options to address issues.	8.NI.C.01 Critique physical and digital procedures that could be implemented to protect electronic data/information.
Clarification Statement: 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.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	MS-PS4-3	CV8.2.1, CV8.3.1, CV8.4.3, CV8.5.3	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
RI.8.7			PE 8.3.3



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Networks & the Internet

Practice(s): 4.4

By end of Grade 8

Standard:
Cybersecurity

6.NI.C.02 Explain the importance of cybersecurity and describe how one method of encryption works.

7.NI.C.02 Identify and explain two or more methods of encryption used to ensure and secure the transmission of information.

8.NI.C.02 Apply multiple methods of encryption to model the secure transmission of data.

Clarification Statement: Encryption can be as simple as letter substitution or as complicated as modern methods used to secure networks and the Internet. Students should encode and decode messages using a variety of encryption methods, and they should understand the different levels of complexity used to hide or secure information. For example, students could secure messages using methods such as Caesar cyphers or steganography (e.g., hiding messages inside a picture or other data). They can also model more complicated methods, such as public key encryption, through unplugged activities.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
7.NS.B.1, 7.NS.B.3	MS-PS4-3	CV8.5.3, CV8.5.4	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Data Analysis

Practice(s): 4.4

By end of Grade 8

**Standard:
Storage**

6.DA.S.01 Represent data using multiple encoding schemes (e.g., images are stored in multiple formats: .jpeg, .png, .gif).

7.DA.S.01 Represent data using multiple encoding schemes (e.g., color names, RGB coding and hexadecimal).

8.DA.S.01 Represent data using multiple encoding schemes (e.g., ASCII, binary).

Clarification Statement: Data representations occur at multiple levels of abstraction, from the physical storage of bits to the arrangement of information into organized formats (e.g., tables). Students should represent the same data in multiple ways. For example, students could represent the same color using binary, RGB values, hex codes (low-level representations), as well as forms understandable by people, including words, symbols, and digital displays of the color (high-level representations).

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
6.NS.D.7, 6.EE.E.2c, 7.NS.B.1, 7.NS.B.1a, 7.NS.B.1c, 7.NS.B.1d, 7.NS.B.1e, 7.NS.B.3		CV8.2.1, CV8.5.3, CV8.5.4	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Data Analysis

Practice(s): 6.3

By end of Grade 8

Standard: Collection, Visualization, & Transformation	6.DA.CVT.01 Explore a variety of computational tools and the content of their data.	7.DA.CVT.01 Collect data using computational tools.	8.DA.CVT.01 Using computational tools, transform collected data to make it more useful and reliable.
	Clarification Statement: As students continue to build on their ability to organize and present data visually to support a claim, they will need to understand when and how to transform data for this purpose. Students should transform data to remove errors, highlight or expose relationships, and/or make it easier for computers to process. The cleaning of data is an important transformation for ensuring consistent format and reducing noise and errors (e.g., removing irrelevant responses in a survey). An example of a transformation that highlights a relationship is representing males and females as percentages of a whole instead of as individual counts.		

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	MS-PS3-1, MS-LS2-1, MS-ESS1-3, MS-ESS2-5, MS-ESS3-2	CV8.4.4, CV8.5.4	5b - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
	SS8.6.3		PE 8.3.3



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Data Analysis

Practice(s): 4.4, 5.3

By end of Grade 8

Standard:
**Inference &
Models**

6.DA.IM.01 Use models and simulations to formulate, refine, and test hypotheses.

7.DA.IM.01 Test and analyze the effects of changing variables while using computational models.

8.DA.IM.01 Refine computational models based on generated data.

Clarification Statement: A model may be a programmed simulation of events or a representation of how various data is related. In order to refine a model, students need to consider which data points are relevant, how data points relate to each other, and if the data is accurate. For example, students may make a prediction about how far a ball will travel based on a table of data related to the height and angle of a track. The students could then test and refine their model by comparing predicted versus actual results and considering whether other factors are relevant (e.g., size and mass of the ball). Additionally, students could refine game mechanics based on test outcomes in order to make the game more balanced or fair.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
7.SP.I.6	MS-ESS2-5, MS-ESS3-2, MS-ETS1-3	CV8.5.3, CV8.5.4	4c - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			PE 8.3.3



Domain: Algorithms & Programming

Practice(s): 4.1, 4.4

By end of Grade 8

Standard: Algorithms	6.AP.A.01 Use existing algorithms in natural language, flowcharts, or pseudocode to solve complex problems.	7.AP.A.01 Select and modify existing algorithms in flowcharts or pseudocode to solve complex problems.	8.AP.A.01 Create flowcharts and pseudocode to design algorithms to solve complex problems.
Clarification Statement: Complex problems are problems that would be difficult for students to solve computationally. Students should use pseudocode and/or flowcharts to organize and sequence an algorithm that addresses a complex problem, even though they may not actually program the solutions. For example, students might express an algorithm that produces a recommendation for purchasing sneakers based on inputs such as size, colors, brand, comfort, and cost. Testing the algorithm with a wide range of inputs and users allows students to refine their recommendation algorithm and to identify other inputs they may have initially excluded.			

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	MS-PS1-2, MS-PS1-4, MS-PS3-3	CV8.4.1, CV8.4.3, CV8.4.4	6c - Creative Communicator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
		FPA8.1.A.1, FPA8.1.A.2, FPA8.1.D.5, FPA8.1.M.4, FPA8.1.D.6	



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Algorithms & Programming

Practice(s): 5.1, 5.2

By end of Grade 8

**Standard:
Variables**

6.AP.V.01 Using grade appropriate content and complexity, create clearly named variables that represent different data types and perform operations on their values.

7.AP.V.01 Using grade appropriate content and complexity, create clearly named variables that represent different data types and perform operations on their values.

8.AP.V.01 Using grade appropriate content and complexity, create clearly named variables that represent different data types and perform operations on their values.

Clarification Statement: A variable is like a container with a name, in which the contents may change, but the name (identifier) does not. When planning and developing programs, students should decide when and how to declare and name new variables. Students should use naming conventions to improve program readability. Examples of operations include adding points to the score, combining user input with words to make a sentence, changing the size of a picture, or adding a name to a list of people.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
6.EE.E.2, 6.EE.E.2a, 6.EE.G.9, 6.EE.F.6, 7.EE.D.4, 7.EE.C.2, 8.EE.D.7, 8.F.E.1	MS-PS1-2, MS-LS2-1, MS-ESS1-3, MS-ETS1-3	CV8.5.3, CV8.5.4	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Algorithms & Programming

Practice(s): 5.1, 5.2

By end of Grade 8

**Standard:
Control**

6.AP.C.01 Using grade appropriate content and complexity, design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.

7.AP.C.01 Using grade appropriate content and complexity, design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.

8.AP.C.01 Using grade appropriate content and complexity, design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.

Clarification Statement: 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, NOT), and nesting conditionals within one another allows the result of one conditional to lead to another. For example, when programming an interactive story, students could use a compound conditional within a loop to unlock a door only if a character has a key AND is touching the door.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
6.EE.E.2, 6.EE.E.4, 6.EE.F.8, 6.EE.G.9, 7.NS.B.3, 8.EE.D.7	MS-ETS1-4		4a - Innovative Designer 5a - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			PE 8.3.3



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Algorithms & Programming

Practice(s): 3.2

By end of Grade 8

Standard: Modularity	6.AP.M.01 Using grade appropriate content and complexity, decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.	7.AP.M.01 Using grade appropriate content and complexity, decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.	8.AP.M.01 Using grade appropriate content and complexity, decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
<p>Clarification Statement: Students should break down problems into subproblems, which can be further broken down to smaller parts. Decomposition facilitates aspects of program development by allowing students to focus on one piece at a time (e.g., getting input from the user, processing the data, and displaying the result to the user). Decomposition also enables different students to work on different parts at the same time. For example, animations can be decomposed into multiple scenes, which can be developed independently.</p>			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
6.EE.E.2b, 6.G.H.1, 6.G.H.4, 7.NS.B.3, 8.EE.D.7, 8.EE.D.8	MS-ETS1-1, MS-ETS1-2, MS-ETS2-2	CV8.3.1, CV8.5.4	5c - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			PE 8.3.3

Domain: Algorithms & Programming
Practice(s): 4.1, 4.3

By end of Grade 8

Standard: Modularity	6.AP.M.02 Using grade appropriate content and complexity, create procedures with parameters to organize code and make it easier to reuse.	7.AP.M.02 Using grade appropriate content and complexity, create procedures with parameters to organize code and make it easier to reuse.	8.AP.M.02 Using grade appropriate content and complexity, create procedures with parameters to organize code and make it easier to reuse.
Clarification Statement: Students should create procedures and/or functions that are used multiple times within a program to repeat groups of instructions. These procedures can be generalized by defining parameters that create different outputs for a wide range of inputs. For example, a procedure to draw a circle involves many instructions, but all of them can be invoked with one instruction, such as “drawCircle.” By adding a radius parameter, the user can easily draw circles of different sizes.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
6.EE.E.2, 6.EE.G.9, 7.NS.B.3, 8.F.E.1		CV8.3.1, CV8.4.4, CV8.5.3, CV8.5.4	5c - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			PE 8.3.3



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Algorithms & Programming

Practice(s): 1.1, 2.3

By end of Grade 8

Standard: Program Development	6.AP.PD.01 Using grade appropriate content and complexity, seek and incorporate feedback from team members and users to refine a solution to a problem.	7.AP.PD.01 Using grade appropriate content and complexity, seek and incorporate feedback from team members and users to refine a solution to a problem.	8.AP.PD.01 Using grade appropriate content and complexity, seek and incorporate feedback from team members and users to refine a solution to a problem.
<p>Clarification Statement: Development teams that employ user-centered design create solutions (e.g., programs and devices) that can have a large societal impact, such as an app that allows people with speech difficulties to translate hard-to-understand pronunciation into understandable language. Students should begin to seek diverse perspectives throughout the design process to improve their computational artifacts. Considerations of the end-user may include usability, accessibility, age-appropriate content, respectful language, user perspective, pronoun use, color contrast, and ease of use.</p>			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	MS-ETS1-1, MS-ETS1-2, MS-ETS1-3, MS-ETS1-4	CV8.2.1, CV8.4.1	7b - Global Collaborator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
W.8.7			PE 8.3.3 HE8.2.1



Domain: Algorithms & Programming

Practice(s): 4.2, 5.2, 7.3

By end of Grade 8

Standard: Program Development	6.AP.PD.02 Incorporate existing code, media, or libraries into original programs and give attribution.	7.AP.PD.02 Incorporate existing code, media, and/or libraries into original programs of increasing complexity and give attribution.	8.AP.PD.02 Incorporate existing code, media, and libraries into original programs of increasing complexity and give attribution.
Clarification Statement: Building on the work of others enables students to produce more interesting and powerful creations. Students should use portions of code, algorithms, and/or digital media in their own programs and websites. At this level, they may also import libraries and connect to application program interfaces (APIs). For example, when creating a side-scrolling game, students may incorporate portions of code that create a realistic jump movement from another person's game, and they may also import Creative Commons-licensed images to use in the background. Students should give attribution to the original creators to acknowledge their contributions.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV8.2.1, CV8.4.4, CV8.5.3, CV8.5.4	6b - Creative Communicator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Algorithms & Programming

Practice(s): 6.1

By end of Grade 8

Standard: Program Development	6.AP.PD.03 Test and refine programs using teacher provided inputs.	7.AP.PD.03 Test and refine programs using a variety of student and peer created inputs.	8.AP.PD.03 Systematically test and refine programs using a range of test cases.
	Clarification Statement: Use cases and test cases are created and analyzed to better meet the needs of users and to evaluate whether programs function as intended. At this level, testing should become a deliberate process that is more iterative, systematic, and proactive than at lower levels. Students should begin to test programs by considering potential errors, such as what will happen if a user enters invalid input (e.g., negative numbers and 0 instead of positive numbers).		

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	MS-ETS1-4, MS-LS4-6		4c - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			PE 8.3.3



Domain: Algorithms & Programming

Practice(s): 7.2

By end of Grade 8

Standard: Program Development	6.AP.PD.04 Using grade appropriate content and complexity, document programs in order to make them easier to follow, test, and debug.	7.AP.PD.04 Using grade appropriate content and complexity, document programs in order to make them easier to follow, test, and debug.	8.AP.PD.04 Using grade appropriate content and complexity, document programs in order to make them easier to follow, test, and debug.
--	--	--	--

Clarification Statement: Documentation allows creators and others to more easily use and understand a program. Students should provide documentation for end users that explains their artifacts and how they function. For example, students could provide a project overview and clear user instructions. They should also incorporate comments in their product and communicate their process using design documents, flowcharts, and presentations.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV8.2.1, CV8.4.1, CV8.4.3, CV8.5.4	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			PE 8.3.3



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Algorithms & Programming

Practice(s): 2.2

By end of Grade 8

Standard: Program Development

6.AP.PD.05 Using a pre-written computational artifact, identify the project timeline tasks necessary for program development.

7.AP.PD.05 Break down tasks and follow an individual timeline when developing a computational artifact.

8.AP.PD.05 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.

Clarification Statement: Collaboration is a common and crucial practice in programming development. Often, many individuals and groups work on the interdependent parts of a project together. Students should assume pre-defined roles within their teams and manage the project workflow using structured timelines. With teacher guidance, they will begin to create collective goals, expectations, and equitable workloads. For example, students may divide the design stage of a game into planning the storyboard, flowchart, and different parts of the game mechanics. They can then distribute tasks and roles among members of the team and assign deadlines.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV8.2.3, CV8.5.2	7c - Global Collaborator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			PE 8.3.3 HE8.2.1



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Impacts of Computing

Practice(s): 7.2

By end of Grade 8

Standard: Culture	6.IC.C.01 Explain how computing impacts people's everyday activities.	7.IC.C.01 Explain how computing impacts innovation in other fields and career opportunities.	8.IC.C.01 Describe impacts associated with computing technologies that affect people's everyday activities and career options.
--	--	---	---

Clarification Statement: Advancements in computer technology are neither wholly positive nor negative. However, the ways that people use computing technologies have tradeoffs. For example, students could compare tradeoffs associated with computing technologies that affect people's everyday activities and career options. Students should consider current events related to broad ideas, including privacy, communication, and automation. For example, driverless cars can increase convenience and reduce accidents, but they are also susceptible to hacking. The emerging industry will reduce the number of taxi and shared-ride drivers, but will create more software engineering and cybersecurity jobs.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	MS-ETS1-1, MS-ETS2-2, MS-PS4-3	CV8.2.1	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
	SS8.3.3	FPA8.3.M.3	



Domain: Impacts of Computing

Practice(s): 1.2

By end of Grade 8

Standard: Culture

6.IC.C.02 Explore issues of bias and accessibility in the design of technologies.

7.IC.C.02 Discuss issues of bias and accessibility in the design of technologies.

8.IC.C.02 Describe issues of bias and accessibility in the design of technologies.

Clarification Statement: 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.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	MS-PS4-3	CV8.2.1	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Impacts of Computing

Practice(s): 2.4, 5.2

By end of Grade 8

Standard: Social Interactions	6.IC.SI.01 Using grade appropriate content and complexity, collaborate using tools to connect with peers when creating a computational artifact.	7.IC.SI.01 Using grade appropriate content and complexity, collaborate using tools to connect with peers when creating a computational artifact.	8.IC.SI.01 Using grade appropriate content and complexity, collaborate using tools to connect with peers when creating a computational artifact.
<p>Clarification Statement: 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). For example, a group of students could combine animations to create a digital community mosaic. They could also solicit feedback from many people through use of online communities and electronic surveys. Collaborating soft skills include an ability to function in teams, an understanding of professional and ethical responsibility, and an ability to communicate effectively. Effective conflict management involves attention to resources, objectives, and identify issues.</p>			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV8.2.2, CV8.2.3, CV8.3.4, CV8.4.4, CV8.5.2, CV8.5.4	7b - Global Collaborator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
	SS8.6.3	FPA8.1.A.4, FPA8.1.T.4, FPA8.1.D.5, FPA8.1.D.6	PE 8.3.3



2019 Wyoming Computer Science Standards

Grade Band: 6-8

Domain: Impacts of Computing

Practice(s): 2.1, 7.3

By end of Grade 8

Standard: Social Interactions	6.IC.SI.02 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.	7.IC.SI.02 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.	8.IC.SI.02 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.
Clarification Statement: Students engage in positive, safe, legal and ethical behavior when using technology and follow school district policy for reporting inappropriate behavior. Students can also describe how they would report inappropriate behavior in an online community and/or to law enforcement. Examples of inappropriate behavior might include sharing or modifying another person's private data, providing inappropriate feedback on another person's project, posting content under another person's name or account, or sharing data without permission.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV8.2.4, CV8.3.4, CV8.4.2, CV8.5.1	2b - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
	SS8.6.3	FPA8.1.A.5, FPA8.4.A.4, FPA8.4.M.1, FPA8.4.T.2	PE 8.3.3 HE6.4.8, HE6.4.9, HE8.4.9, HE8.4.10



Domain: Impacts of Computing

Practice(s): 7.2

By end of Grade 8

Standard: Safety, Law, & Ethics	6.IC.SLE.01 Using grade appropriate content and complexity, describe tradeoffs between allowing information to be public and keeping information private and secure.	7.IC.SLE.01 Using grade appropriate content and complexity, describe tradeoffs between allowing information to be public and keeping information private and secure.	8.IC.SLE.01 Using grade appropriate content and complexity, describe tradeoffs between allowing information to be public and keeping information private and secure.
Clarification Statement: 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.			

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV8.2.1, CV8.2.4	2c - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH

Domain: Impacts of Computing

Practice(s): 1.1, 7.2

By end of Grade 8

Standard: Safety, Law, & Ethics	6.IC.SLE.02 Using grade level appropriate content and complexity, discuss the legal, social, and ethical impacts associated with software development and use, including both positive and malicious intent.	7.IC.SLE.02 Using grade level appropriate content and complexity, discuss the legal, social, and ethical impacts associated with software development and use, including both positive and malicious intent.	8.IC.SLE.02 Using grade level appropriate content and complexity, discuss the legal, social, and ethical impacts associated with software development and use, including both positive and malicious intent.
Clarification Statement: Ethics involves a focus on real-world applications of emerging technology, diverse academic perspectives, discussing existing industry standards for use as ethical guidelines, and developing systematic methods to analyze societal issues. Examples of positive impacts could include writing software or utilities to improve communication for people who have a disability, writing an application that manages money for a bank, or software that handles healthcare records. Examples of negative impacts could include distributing a virus, or writing backdoor code, malware, or ransomware.			

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV8.2.1, CV8.2.4, CV8.4.2	2a - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH

Performance Level Descriptors (PLDs)

Grade Band: 6-8

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Devices: 8.CS.D.01 Recommend improvements to the design of computing devices based on an analysis of how a variety of users interact with the device.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - understands the needs of the users, but is unable to analyze, and/or - describes the parts of computing devices, but cannot recommend improvements to the design. 	<ul style="list-style-type: none"> - analyzes the needs of the users. - recommends improvements to the design of computing devices based on that analysis. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., recommend improvements to the design in more than one area (input, output, processing, storage) or group (special populations)).
Hardware & Software: 8.CS.HS.01 Design and refine a project that combines hardware and software components to collect and exchange data.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - describes how hardware and software components collect and exchange data, but cannot design a project, and/or - creates a project that combines hardware and software components to collect and exchange data but cannot refine. 	<ul style="list-style-type: none"> - designs a project that combines hardware and software components to collect and exchange data. - refines a project that combines hardware and software components to collect and exchange data. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., design a project that combines hardware and software components to collect and exchange data that affects the world around them, refine a project multiple times that combines hardware and software components to collect and exchange data to address real world usage).

Performance Level Descriptors (PLDs)

Grade Band: 6-8

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Troubleshooting: 8.CS.T.01 Systematically identify, resolve, and document increasingly complex software and hardware problems with computing devices and their components.	provides little to no evidence in addressing the expectation(s).	can do some of the following: - identify software problems with computing devices and their components, - identify hardware problems with computing devices and their components, - resolve software problems with computing devices and their components, - resolve hardware problems with computing devices and their components, - document software problems with computing devices and their components, - document hardware problems with computing devices and their components.	can systematically: - identify software problems with computing devices and their components, - identify hardware problems with computing devices and their components, - resolve software problems with computing devices and their components, - resolve hardware problems with computing devices and their components, - document software problems with computing devices and their components, - document hardware problems with computing devices and their components.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., systematically assists others with hardware or software problems, creates a detailed troubleshooting document or tutorial, comes up with novel solutions).

Performance Level Descriptors (PLDs)

Grade Band: 6-8

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Network Communication & Organization: 8.NI.NCO.01 Model the role of protocols in transmitting data across networks and the internet (e.g. explain protocols and their importance to data transmission; model how packets are broken down into smaller pieces and how they are delivered).	provides little to no evidence in addressing the expectation(s).	- identifies protocols used in transmitting data across networks and the internet, and/or - explains the role of protocols in transmitting data across networks and the internet.	- models the role of protocols in transmitting data across networks and the internet.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., research and compare/contrast multiple network protocols).
Cybersecurity: 8.NI.C.01 Using grade appropriate content and complexity, create programs that use variables to store and modify data. Cybersecurity: 8.NI.C.02 Apply multiple methods of encryption to model the secure transmission of data.	provides little to no evidence in addressing the expectation(s).	- lists physical and digital procedures that could be implemented to protect electronic data/ information, and/or - describes multiple methods of encryption used to secure data.	- critiques physical and digital procedures that could be implemented to protect electronic data/information. - applies multiple methods of encryption to model the secure transmission of data.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., explain the impacts of hacking, ransomware, scams, and ethical/legal concerns; compare the advantages and disadvantages of multiple methods of encryption to model the secure transmission of information).

Performance Level Descriptors (PLDs)

Grade Band: 6-8

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Storage: 8.DA.S.01 Represent data using multiple encoding schemes (e.g., ASCII, binary).	provides little to no evidence in addressing the expectation(s).	- recognizes data is stored in multiple encoding schemes.	- represents data using multiple encoding schemes.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., convert data between multiple encoding schemes; ASCII to binary, hex to rgb).
Collection, Visualization, & Transformation: 8.DA.CVT.01 Using computational tools, transform collected data to make it more useful and reliable.	provides little to no evidence in addressing the expectation(s).	- explores a variety of computational tools and the content of their data. - uses computational tools to collect data.	determines appropriate computational tools to: - transform data to remove errors. - highlight or expose relationships in the data.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., error checking input during data collection process, export data to another format).
Inference & Models: 8.DA.IM.01 Refine computational models based on generated data.	provides little to no evidence in addressing the expectation(s).	- uses models and simulations to formulate, refine, and test hypotheses, and/or - tests and analyzes the effects of changing variables while using computational models.	- refines computational models based on generated data.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., make multiple refinements).

Performance Level Descriptors (PLDs)

Grade Band: 6-8

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Algorithms: 8.AP.A.01 Create flowcharts and pseudocode to design algorithms to solve complex problems.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - uses flowcharts to modify existing algorithms, and/or - uses pseudocode to modify existing algorithms, and/or - uses natural language to modify existing algorithms. 	<ul style="list-style-type: none"> - creates flowcharts to design algorithms to solve complex problems. - writes pseudocode to design algorithms to solve complex problems. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., design algorithms to solve complex problems in multiple ways and determine and use the most effective planning tool).
Variables: 8.AP.V.01 Using grade appropriate content and complexity, create clearly named variables that represent different data types and perform operations on their values.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - recognizes that variables can represent different data types, and/or - can create a variable, and/or - can perform operations on the values of variables. 	<ul style="list-style-type: none"> - clearly names variables. - creates variables that represent different data types. - performs operations on the values of variables. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., explain types of errors that can occur if improper data types are used in operations, understand structures or classes can contain multiple data types).
Control: 8.AP.C.01 Using grade appropriate content and complexity, design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.	provides little to no evidence in addressing the expectation(s).	designs and iteratively develops programs that: <ul style="list-style-type: none"> - use simple loops. - use simple conditionals. 	designs and iteratively develops programs that include: <ul style="list-style-type: none"> - nested loops. - compound conditionals. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., multiple examples of nested loops and compound conditions in a program, evidence of efficient code, clear documentation).

Performance Level Descriptors (PLDs)

Grade Band: 6-8

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
<p>Modularity: 8.AP.M.01 Using grade appropriate content and complexity, decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.</p> <p>Modularity: 8.AP.M.02 Using grade appropriate content and complexity, create procedures with parameters to organize code and make it easier to reuse.</p>	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - recognizes the inefficiency of repetition in programming, and/or - recognizes the organizational, readability and labor-saving advantages of code reuse. 	<ul style="list-style-type: none"> - decomposes problems and subproblems into parts. - creates procedures with parameters. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., create procedures with multiple parameters and/or return values).

Performance Level Descriptors (PLDs)

Grade Band: 6-8

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
<p>Program Development: 8.AP.PD.01 Using grade appropriate content and complexity, seek and incorporate feedback from team members and users to refine a solution to a problem.</p> <p>Program Development: 8.AP.PD.02 Incorporate existing code, media, and libraries into original programs of increasing complexity and give attribution.</p> <p>Program Development: 8.AP.PD.03 Systematically test and refine programs using a range of test cases.</p> <p>Program Development: 8.AP.PD.04 Using grade appropriate content and complexity, document programs in order to make them easier to follow, test, and debug.</p>	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - recognizes the advantage of using existing code. - recognizes reasons for testing and refining programs. - recognizes the advantage of documenting programs. - recognizes the role of using feedback. 	<ul style="list-style-type: none"> - incorporates existing code, media, and libraries into original programs. - systematically tests and refines programs. - documents programs. - seeks and incorporates feedback. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., seek open source libraries to include in their program, seek feedback from a wide audience).
<p>Program Development: 8.AP.PD.05 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.</p>	provides little to no evidence in addressing the expectation(s).	<p>using a pre-written computational artifact:</p> <ul style="list-style-type: none"> - identifies the project timeline tasks necessary for program development. - breaks down tasks and follows an individual timeline when developing a computational artifact. 	<p>when collaboratively developing computational artifacts:</p> <ul style="list-style-type: none"> - distributes tasks. - maintains a project timeline. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., adjust the timeline and redistribute tasks to meet the deadline).

Performance Level Descriptors (PLDs)

Grade Band: 6-8

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
<p>Culture: 8.IC.C.01 Describe impacts associated with computing technologies that affect people's everyday activities and career options.</p> <p>Culture: 8.IC.C.02 Describe issues of bias and accessibility in the design of technologies.</p>	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - lists computing technologies that affect people's everyday activities, and/or - lists computing technologies that affect people's career options, and/or - identifies an accessibility issue related to technology. 	<ul style="list-style-type: none"> - describes impacts associated with computing technologies that affect people's everyday activities. - describes impacts associated with computing technologies that affect people's career options. - describes issues of bias and accessibility in the design of technologies. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., devise solutions to solve issues of bias in accessibility, reduce negative impacts of computing technology in everyday life).
<p>Social Interactions: 8.IC.SI.01 Using grade appropriate content and complexity, collaborate using tools to connect with peers when creating a computational artifact.</p> <p>Social Interactions: 8.IC.SI.02 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.</p>	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - collaborates with peers using a tool in an attempt to create a computational artifact. - intermittently collaborates and behaves within an online community. 	<ul style="list-style-type: none"> - collaborates using tools to connect with peers when creating a computational artifact. - practices grade-level appropriate behavior and responsibilities while participating in an online community. - identifies and reports inappropriate behavior while participating in an online community, when applicable. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., moderate, model appropriate behavior, and facilitate discussions in an online community).

Performance Level Descriptors (PLDs)

Grade Band: 6-8

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
<p>Safety, Law, & Ethics: 8.IC.SLE.01 Using grade appropriate content and complexity, describe tradeoffs between allowing information to be public and keeping information private and secure.</p> <p>Safety, Law, & Ethics: 8.IC.SLE.02 Using grade appropriate content and complexity, describe tradeoffs between allowing information to be public and keeping information private and secure.</p>	<p>provides little to no evidence in addressing the expectation(s).</p>	<p>- lists reasons for allowing information to be public and keeping information private and secure, and/or With regard to positive and/or malicious intent can:</p> <ul style="list-style-type: none"> - name the legal impacts associated with software development and use, - name the social impacts associated with software development and use, - name the ethical impacts associated with software development and use. 	<p>- describes tradeoffs between allowing information to be public and keeping information private and secure, and With regard to positive and malicious intent:</p> <ul style="list-style-type: none"> - discusses the legal impacts associated with software development and use, - discusses the social impacts associated with software development and use, - discusses the ethical impacts associated with software development and use. 	<p>demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., research and report on current legal, social, and ethical worldwide trends in software development; construct an argument for or against the use of personal data by commercial entities or government).</p>

Computer Science | 9-12 Introduction

In high school, students will continue to develop their knowledge of computing systems, their components, and how systems interact. Students will use their understanding about the basic principles of computation, that algorithms describe a step-by-step solution to a problem, that programs are algorithms written in a language that a computer can understand, and that the solution to many problems can be described as a program. A solid foundation of algebraic concepts is important for success in high school computer science courses. Students will expand their ability to identify patterns and create algorithms that can model the observed patterns.

By the end of 12th grade, students can:

- Create a computer program using sequencing, selection, and iteration
- Decompose complex problems into smaller, more manageable sections
- Use tools of coding to create, debug, and document the evolution of an artifact
- Compare and contrast trade-offs in programming techniques
- Develop complex computer program individually and as part of a group
- Recognize how various components of a complex computing system work together
- Use tools to analyze data and know how data is stored
- Explain how cybersecurity issues affect networks and the internet
- Justify how proliferation of computing affects privacy, rights, opportunities, and responsibility

The high school standards are organized into 2 levels. Mostly, Level 1 is intended to be at the introductory level, and Level 2 reaches at a deeper level.

WYOMING 2019 COMPUTER SCIENCE DOMAINS & STANDARDS

Computing Systems	Networks & The Internet	Data Analysis	Algorithms & Programming	Impacts of Computing
CS.D—Devices	NI.NCO—Network Communication & Organization	DA.S—Storage	AP.A—Algorithms	IC.C—Culture
CS.HS—Hardware & Software	NI.C—Cybersecurity	DA.CVT—Collection, Visualization, & Transformation	AP.V—Variables	IC.SI—Social Interactions
CS.T—Troubleshooting		DA.IM—Inference & Models	AP.C—Control	IC.SLE—Safety, Law, & Ethics
			AP.M—Modularity	
			AP.PD—Program Development	

9-12 Computer Science Practices

There are seven (7) CS Practices that are to be embedded in curriculum and instruction as the standards and benchmarks are taught and measured. The seven (7) CS Practices are listed below, and are more deeply explored on the next several pages. For each grade-band, only the CS Practices that relate are in black text and the others are grayed so the reader can still see them as a set, but will know which ones apply to that particular set of standards.

Practice 1. Fostering an Inclusive Computing Culture

Practice 2. Collaborating Around Computing

Practice 3. Recognizing and Defining Computational Problems

Practice 4. Developing and Using Abstractions

Practice 5. Creating Computational Artifacts

Practice 6. Testing and Refining Computational Artifacts

Practice 7. Communicating About Computing

DESCRIPTION OF 9-12 COMPUTER SCIENCE (CS) PRACTICES

CS Practice 1. Fostering an Inclusive Computing Culture

Overview: 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.

At all grade levels, students should recognize that the choices people make when they create artifacts are based on personal interests, experiences, and needs. Young learners should begin to differentiate their technology preferences from the technology preferences of others. Initially, students should be presented with perspectives from people with different backgrounds, ability levels, and points of view. As students progress, they should independently seek diverse perspectives throughout the design process for the purpose of improving their computational artifacts. Students who are well-versed in fostering an inclusive computing culture should be able to differentiate backgrounds and skill sets and know when to call upon others, such as to seek out knowledge about potential end users or intentionally seek input from people with diverse backgrounds.

1.2 Address the needs of diverse end users during the design process to produce artifacts with broad accessibility and usability.

At any level, students should recognize that users of technology have different needs and preferences and that not everyone chooses to use, or is able to use, the same technology products. For example, young learners, with teacher guidance, might compare a touchpad and a mouse to examine differences in usability. As students progress, they should consider the preferences of people

who might use their products. Students should be able to evaluate the accessibility of a product to a broad group of end users, such as people with various disabilities. For example, they may notice that allowing an end user to change font sizes and colors will make an interface usable for people with low vision. At the higher grades, students should become aware of professionally accepted accessibility standards and should be able to evaluate computational artifacts for accessibility. Students should also begin to identify potential bias during the design process to maximize accessibility in product design. For example, they can test an app and recommend to its designers that it respond to verbal commands to accommodate users who are blind or have physical disabilities.

1.3 Employ self- and peer-advocacy to address bias in interactions, product design, and development methods.

After students have experience identifying diverse perspectives and including unique perspectives (P1.1), they should begin to employ self-advocacy strategies, such as speaking for themselves if their needs are not met. As students progress, they should advocate for their peers when accommodations, such as an assistive-technology peripheral device, are needed for someone to use a computational artifact. Eventually, students should regularly advocate for both themselves and others.

CS Practice 2. Collaborating Around Computing

Overview: 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 Cultivate working relationships with individuals possessing diverse perspectives, skills, and personalities.

At any grade level, students should work collaboratively with others. Early on, they should learn strategies for working with team members who possess varying individual strengths. For example, with teacher support, students should begin to give each team member opportunities to contribute and to work with each other as co-learners. Eventually, students should become more sophisticated at applying strategies for mutual encouragement and support. They should express their ideas with logical reasoning and find ways to reconcile differences cooperatively. For example, when they disagree, they should ask others to explain their reasoning and work together to make respectful, mutual decisions. As they progress, students should use methods for giving all group members a chance to participate. Older students should strive to improve team efficiency and effectiveness by regularly evaluating group dynamics. They should use multiple strategies to make team dynamics more productive. For example, they can ask for the opinions of quieter team members, minimize interruptions by more talkative members, and give individuals credit for their ideas and their work.

- 2.2 Create team norms, expectations, and equitable workloads to increase efficiency and effectiveness.

After students have had experience cultivating working relationships within teams (P2.1), they should gain experience working in particular team roles. Early on, teachers may help guide this process by providing collaborative structures. For example, students may take turns in different roles on the project, such as note taker, facilitator, or “driver” of the computer. As students progress, they should become less dependent on the teacher assigning roles and become more adept at assigning roles within their teams. For example, they should decide together how to take turns in different roles. Eventually, students should independently organize their own teams and create common goals, expectations, and equitable workloads. They should also manage project workflow using agendas and timelines and should evaluate workflow to

identify areas for improvement.

- 2.3 Solicit and incorporate feedback from, and provide constructive feedback to, team members and other stakeholders.

At any level, students should ask questions of others and listen to their opinions. Early on, with teacher scaffolding, students should seek help and share ideas to achieve a particular purpose. As they progress in school, students should provide and receive feedback related to computing in constructive ways. For example, pair programming is a collaborative process that promotes giving and receiving feedback. Older students should engage in active listening by using questioning skills and should respond empathetically to others. As they progress, students should be able to receive feedback from multiple peers and should be able to differentiate opinions. Eventually, students should seek contributors from various environments. These contributors may include end users, experts, or general audiences from online forums.

- 2.4 Evaluate and select technological tools that can be used to collaborate on a project.

At any level, students should be able to use tools and methods for collaboration on a project. For example, in the early grades, students could collaboratively brainstorm by writing on a white-board. As students progress, they should use technological collaboration tools to manage team-work, such as knowledge-sharing tools and online project spaces. They should also begin to make decisions about which tools would be best to use and when to use them. Eventually, students should use different collaborative tools and methods to solicit input from not only team members and classmates but also others, such as participants in online forums or local communities.

CS Practice 3. Recognizing and Defining Computational Problems

Overview: 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.

At any level, students should be able to identify problems that have been solved computationally. For example, young students can discuss a technology that has changed the world, such as email or mobile phones. As they progress, they should ask clarifying questions to understand whether a problem or part of a problem can be solved using a computational approach. For example, identify real-world problems that span multiple disciplines, such as increasing bike safety with new helmet technology, and can be solved computationally.

3.2 Decompose complex real-world problems into manageable sub-problems that could integrate existing solutions or procedures.

At any grade level, students should be able to break problems down into their component parts. In the early grade levels, students should focus on breaking down simple problems. For example, in a visual programming environment, students could break down (or decompose) the steps needed to draw a shape. As students progress, they should decompose larger problems into manageable smaller problems. For example, young students may think of an animation as multiple scenes and thus create each scene independently. Students can also break down a program into subgoals: getting input from the user, processing the data, and displaying the result to the user. Eventually, as students encounter complex real-world problems that span multiple disciplines or social systems, they should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist. For example, students could create an app to solve a community problem that connects to an online database through an application programming interface (API).

3.3 Evaluate whether it is appropriate and feasible to solve a problem computationally.

After students have had some experience breaking problems down (P3.2) and

identifying subproblems that can be solved computationally (P3.1), they should begin to evaluate whether a computational solution is the most appropriate solution for a particular problem. For example, students might question whether using a computer to determine whether someone is telling the truth would be advantageous. As students progress, they should systematically evaluate the feasibility of using computational tools to solve given problems or subproblems, such as through a cost-benefit analysis. Eventually, students should include more factors in their evaluations, such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns.

CS Practice 4. Developing and Using Abstractions

Overview: 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.

Students at all grade levels should be able to recognize patterns. Young learners should be able to identify and describe repeated sequences in data or code through analogy to visual patterns or physical sequences of objects. As they progress, students should identify patterns as opportunities for abstraction, such as recognizing repeated patterns of code that could be more efficiently implemented as a loop. Eventually, students should extract common features from more complex phenomena or processes. For example, students should be able to identify common features in multiple segments of code and substitute a single segment that uses variables to account for the differences. In a procedure, the variables would take the form of parameters. When working with data, students should be able to identify important aspects and find patterns in related data sets such as crop output, fertilization methods, and climate conditions.

4.2 Evaluate existing technological functionalities and incorporate them into new designs.

At all levels, students should be able to use well-defined abstractions that hide complexity. Just as a car hides operating details, such as the mechanics of the engine, a computer program's "move" command relies on hidden details that cause an object to change location on the screen. As they progress, students should incorporate predefined functions into their designs, understanding that they do not need to know the underlying implementation details of the abstractions that they use. Eventually, students should understand the advantages of, and be comfortable using, existing functionalities (abstractions) including technological resources created by other people, such as libraries and application programming interfaces (APIs). Students should be able to evaluate existing abstractions to determine which should be incorporated into designs and how they should be incorporated. For example, students could build powerful apps by incorporating existing services, such as online databases that return geolocation coordinates of street names or food nutrition information.

4.3 Create modules and develop points of interaction that can apply to multiple situations and reduce complexity.

After students have had some experience identifying patterns (P4.1), decomposing problems (P3.2), using abstractions (P4.2), and taking advantage of existing resources (P4.2), they should begin to develop their own abstractions. As they progress, students should take advantage of opportunities to develop generalizable modules. For example, students could write more efficient programs by designing procedures that are used multiple times in the program. These procedures can be generalized by defining parameters that create different outputs for a wide range of inputs. Later on, students should be able to design systems of interacting modules, each with a well-defined role, that coordinate to accomplish a common goal. Within an object-oriented programming context, module design may include defining interactions among objects. At this stage, these modules, which combine both data and procedures, can be designed and documented for reuse in other

programs. Additionally, students can design points of interaction, such as a simple user interface, either text or graphical, that reduces the complexity of a solution and hides lower-level implementation details.

4.4 Model phenomena and processes and simulate systems to understand and evaluate potential outcomes.

Students at all grade levels should be able to represent patterns, processes, or phenomena. With guidance, young students can draw pictures to describe a simple pattern, such as sunrise and sunset, or show the stages in a process, such as brushing your teeth. They can also create an animation to model a phenomenon, such as evaporation. As they progress, students should understand that computers can model real-world phenomena, and they should use existing computer simulations to learn about real-world systems. For example, they may use a preprogrammed model to explore how parameters affect a system, such as how rapidly a disease spreads. Older students should model phenomena as systems, with rules governing the interactions within the system. Students should analyze and evaluate these models against real-world observations. For example, students might create a simple producer–consumer ecosystem model using a programming tool. Eventually, they could progress to creating more complex and realistic interactions between species, such as predation, competition, or symbiosis, and evaluate the model based on data gathered from nature.

CS Practice 5. Creating Computational Artifacts

Overview: 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.

At any grade level, students should participate in project planning and the creation of brainstorming documents. The youngest students may do so with the help of teachers. With scaffolding, students should gain greater independence and sophistication in the planning, design, and evaluation of artifacts. As learning progresses, students should systematically plan the development of a program or artifact and intentionally apply computational techniques, such as decomposition and abstraction, along with knowledge about existing approaches to artifact design. Students should be capable of reflecting on and, if necessary, modifying the plan to accommodate end goals.

5.2 Create a computational artifact for practical intent, personal expression, or to address a societal issue.

Students at all grade levels should develop artifacts in response to a task or a computational problem. At the earliest grade levels, students should be able to choose from a set of given commands to create simple animated stories or solve pre-existing problems. Younger students should focus on artifacts of personal importance. As they progress, student expressions should become more complex and of increasingly broader significance. Eventually, students should engage in independent, systematic use of design processes to create artifacts that solve problems with social significance by seeking input from broad audiences.

5.3 Modify an existing artifact to improve or customize it.

At all grade levels, students should be able to examine existing artifacts to understand what they do. As they progress, students should attempt to use existing solutions to accomplish a desired goal. For example, students could attach a programmable light sensor to a physical artifact they have created to make it respond to light. Later on, they should modify or remix parts of existing programs to develop something new or to add more advanced features and

complexity. For example, students could modify prewritten code from a single-player game to create a two-player game with slightly different rules.

CS Practice 6. Testing and Refining Computational Artifacts

Overview: 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 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.

At any grade level, students should be able to compare results to intended outcomes. Young students should verify whether given criteria and constraints have been met. As students progress, they should test computational artifacts by considering potential errors, such as what will happen if a user enters invalid input. Eventually, testing should become a deliberate process that is more iterative, systematic, and proactive. Older students should be able to anticipate errors and use that knowledge to drive development. For example, students can test their program with inputs associated with all potential scenarios.

6.2 Identify and fix errors using a systematic process.

At any grade level, students should be able to identify and fix errors in programs (debugging) and use strategies to solve problems with computing systems (troubleshooting). Young students could use trial and error to fix simple errors. For example, a student may try reordering the sequence of commands in a program. In a hardware context, students could try to fix a device by resetting it or checking whether it is connected to a network. As students progress, they should become more adept at debugging programs and begin to consider logic errors: cases in which a program works, but not as desired. In this way, students will examine and correct their own thinking. For

example, they might step through their program, line by line, to identify a loop that does not terminate as expected. Eventually, older students should progress to using more complex strategies for identifying and fixing errors, such as printing the value of a counter variable while a loop is running to determine how many times the loop runs.

6.3 Evaluate and refine a computational artifact multiple times to enhance its performance, reliability, usability, and accessibility.

After students have gained experience testing (P6.2), debugging, and revising (P6.1), they should begin to evaluate and refine their computational artifacts. As students progress, the process of evaluation and refinement should focus on improving performance and reliability. For example, students could observe a robot in a variety of lighting conditions to determine that a light sensor should be less sensitive. Later on, evaluation and refinement should become an iterative process that also encompasses making artifacts more usable and accessible (P1.2). For example, students can incorporate feedback from a variety of end users to help guide the size and placement of menus and buttons in a user interface.

CS Practice 7. Communicating About Computing

Overview: 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.

At any grade level, students should be able to refer to data when communicating an idea. Early on, students should, with guidance, present basic data through the use of visual representations, such as storyboards,

flowcharts, and graphs. As students progress, they should work with larger data sets and organize the data in those larger sets to make interpreting and communicating it to others easier, such as through the creation of basic data representations. Eventually, students should be able to select relevant data from large or complex data sets in support of a claim or to communicate the information in a more sophisticated manner.

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

At any grade level, students should be able to talk about choices they make while designing a computational artifact. Early on, they should use language that articulates what they are doing and identifies devices and concepts they are using with correct terminology (e.g., program, input, and debug). Younger students should identify the goals and expected outcomes of their solutions. Along the way, students should provide documentation for end users that explains their artifacts and how they function, and they should both give and receive feedback. For example, students could provide a project overview and ask for input from users. As students progress, they should incorporate clear comments in their product and document their process using text, graphics, presentations, and demonstrations.

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

All students should be able to explain the concepts of ownership and sharing. Early on, students should apply these concepts to computational ideas and creations. They should identify instances of remixing, when ideas are borrowed and iterated upon, and give proper attribution. They should also recognize the contributions of collaborators. Eventually, students should consider common licenses that place limitations or restrictions on the use of computational artifacts. For example, a downloaded image may have restrictions that prohibit modification of an image or using it for commercial purposes.



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Computing Systems

Practice(s): 4.1

By end of Grade 12

Standard: Devices	L1.CS.D.01 Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.
Clarification Statement:	Computing devices are often integrated with other systems, including biological, mechanical, and social systems. A medical device can be embedded inside a person to monitor and regulate his or her health, a hearing aid (a type of assistive device) can filter out certain frequencies and magnify others, a monitoring device installed in a motor vehicle can track a person's driving patterns and habits, and a facial recognition device can be integrated into a security system to identify a person. The creation of integrated or embedded systems is not an expectation at this level. Students might select an embedded device such as a car stereo, identify the types of data (e.g., radio station presets, volume level) and procedures (e.g., increase volume, store/recall saved station, mute) it includes, and explain how the implementation details are hidden from the user.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.2.1, CV12.5.2	5c - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH

Domain: Computing Systems
Practice(s): Level 1: 4.1; Level 2: 4.1, 7.2
By end of Grade 12

Standard: Hardware & Software	L1.CS.HS.01 Explain the interactions between application software, system software, and hardware layers.	L2.CS.HS.01 Categorize the roles of operating system software.
Clarification Statement:	Level 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. For example, text editing software interacts with the operating system to receive input from the keyboard, convert the input to bits for storage, and interpret the bits as readable text to display on the monitor. System software is used on many different types of devices, such as smart TVs, assistive devices, virtual components, cloud components, and drones. For example, students may explore the progression from voltage to binary signal to logic gates to adders and so on. Knowledge of specific, advanced terms for computer architecture, such as BIOS, kernel, or bus, is not expected at this level.	Level 2: Examples of roles could include memory management, data storage/retrieval, process management, and access control.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L1—CV12.2.1, CV12.5.2	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: **9-12**

Domain: Computing Systems

Practice(s): Level 1: 6.1, 6.2; Level 2: 7.2

By end of Grade 12

Standard: Troubleshooting	L1.CS.T.01 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and resolve errors.	L2.CS.T.01 Identify how hardware components facilitate logic, input, output, and storage in computing systems, and their common malfunctions.
Clarification Statement:	Level 1: Systematic troubleshooting strategies could include eliminating variables, gathering background information, reproducing the problem, converging on the problem, looking at past documentation, researching, etc. Examples of guidelines could include a flow chart, a job aid for a help desk employee, or an expert system.	Level 2: Examples of components could include logic gates, IO pins, memory, graphics card, CPU, hard drive, internal drive, and motherboard.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	L1—HS-ETS1-2	L1—CV12.2.1, CV12.4.1, CV12.4.3, CV12.4.4, CV12.5.2 L2—CV12.4.3	L1 & L2—3d - Knowledge Constructor
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Networks & the Internet

Practice(s): Level 1: 4.1, 7.2; Level 2: 7.2

By end of Grade 12

Standard: Network Communication & Organization	L1.NI.NCO.01 Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing.	L2.NI.NCO.01 Describe the issues that impact network functionality (e.g., bandwidth, load, latency, topology).
Clarification Statement:	Level 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.	

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L1—CV12.2.1, CV12.4.3, CV12.5.2 L2—CV12.2.1,	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH

Domain: Networks & the Internet

Practice(s): Level 1 & 2: 7.2

By end of Grade 12

Standard: Cybersecurity	L1.NI.C.01 Give examples to illustrate how sensitive data can be affected by malware and other attacks.	L2.NI.C.01 Compare ways software developers protect devices and information from unauthorized access.
Clarification Statement:	Level 1: Network security depends on a combination of hardware, software, and practices that control access to data and systems. The needs of users and the sensitivity of data determine the level of security implemented. Potential security problems, such as denial-of-service attacks, ransomware, viruses, worms, spyware, and phishing, present threats to sensitive data. Students might reflect on case studies or current events in which governments or organizations experienced data leaks or data loss as a result of these types of attacks.	Level 2: Examples of security concerns to consider could include encryption and authentication strategies, secure coding, and safeguarding keys.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L1—CV12.2.1, CV12.5.2 L2—CV12.2.1, CV12.3.3	L1—2d - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Networks & the Internet

Practice(s): 3.3

By end of Grade 12

Standard: Cybersecurity	L1.NI.C.02 Recommend cybersecurity measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.
Clarification Statement:	Level 1: Security measures may include physical security tokens, two-factor authentication, and biometric verification. Potential security problems, such as denial-of-service attacks, ransomware, viruses, worms, spyware, and phishing, exemplify why sensitive data should be securely stored and transmitted. The timely and reliable access to data and information services by authorized users, referred to as availability, is ensured through adequate bandwidth, backups, and other measures. Students should systematically evaluate the feasibility of using computational tools to solve given problems or subproblems, such as through a cost-benefit analysis. Eventually, students should include more factors in their evaluations, such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.2.1, CV12.3.3, CV12.3.4, CV12.5.2	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Networks & the Internet

Practice(s): 6.3

By end of Grade 12

Standard: Cybersecurity	L1.NI.C.03 Compare various security measures, considering trade-offs between the usability and security of a computing system.
Clarification Statement:	Level 1: Security measures may include physical security tokens, two-factor authentication, and biometric verification, but choosing security measures involves tradeoffs between the usability and security of the system. The needs of users and the sensitivity of data determine the level of security implemented. Students might discuss computer security policies in place at the local level that present a tradeoff between usability and security, such as a web filter that prevents access to many educational sites but keeps the campus network safe.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.2.1, CV12.3.3, CV12.5.2	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Networks & the Internet

Practice(s): 7.2

By end of Grade 12

Standard: Cybersecurity	L1.NI.C.04 Explain trade-offs when selecting and implementing cybersecurity recommendations.
Clarification Statement:	Level 1: Network security depends on a combination of hardware, software, and practices that control access to data and systems. The needs of users and the sensitivity of data determine the level of security implemented. Every security measure involves tradeoffs between the accessibility and security of the system. Students should be able to describe, justify, and document choices they make using terminology appropriate for the intended audience and purpose. Students could debate issues from the perspective of diverse audiences, including individuals, corporations, privacy advocates, security experts, and government.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.2.1, CV12.3.3	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Data Analysis

Practice(s): 4.1

By end of Grade 12

Standard: Storage	L1.DA.S.01 Translate between different bit representations of real-world phenomena, such as characters, numbers, and images.
Clarification Statement:	Level 1: For example, convert hexadecimal color codes to decimal percentages, ASCII/Unicode representation, and logic gates.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.2.1	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
RI.9-10.7			



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Data Analysis

Practice(s): 3.3

By end of Grade 12

Standard: Storage	L1.DA.S.02 Evaluate the trade-offs in how data elements are organized and where data is stored.
Clarification Statement:	Level 1: People make choices about how data elements are organized and where data is stored. These choices affect cost, speed, reliability, accessibility, privacy, and integrity. Students should evaluate whether a chosen solution is most appropriate for a particular problem. Students might consider the cost, speed, reliability, accessibility, privacy, and integrity tradeoffs between storing photo data on a mobile device versus in the cloud.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.2.1, CV12.3.3	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH

Domain: Data Analysis

Practice(s): Level 1: 4.4; Level 2: 4.1, 7.1

By end of Grade 12

Standard: Collection, Visualization, & Transformation	L1.DA.CVT.01 Create interactive data representations using software tools to help others better understand real-world phenomena (e.g., paper surveys and online data sets).	L2.DA.CVT.01 Use data analysis tools and techniques to identify patterns in data representing complex systems.
Clarification Statement:	Level 1: People transform, generalize, simplify, and present large data sets in different ways to influence how other people interpret and understand the underlying information. Examples include visualization, aggregation, rearrangement, and application of mathematical operations. People use software tools or programming to create powerful, interactive data visualizations and perform a range of mathematical operations to transform and analyze data. Students should model phenomena as systems, with rules governing the interactions within the system and evaluate these models against real-world observations. For example, flocking behaviors, queueing, or life cycles. Google Fusion Tables can provide access to data visualization online.	Level 2: For example, identify trends in a dataset representing social media interactions, movie reviews, or shopping patterns.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
L1 —S.IC.E.4, S.IC.E.5 L2 —F.TF.I.5, F.LE.F.1, F.IF.B.5, A.CED.G.1, A.CED.G.2, A.CED.G.3	L1 —HS-ETS1-1, HS-ESS3-5, HS-ESS3-6, HS-ETS1-1, HS-ETS1-4 L2 —HS-ESS3-6, HS-ESS3-5, HS-ESS3-3	L1 —CV12.2.1, CV12.3.1, CV12.5.1, CV12.5.2, CV12.5.4 L2 —CV12.3.2	L1 —6c - Creative Communicator L2 —5b - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
	L1 —SS12.5.1, SS12.5.1.a		



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Data Analysis

Practice(s): 7.1, 7.2

By end of Grade 12

Standard: Collection, Visualization, & Transformation	L2.DA.CVT.02 Select data collection tools and techniques, and use them to generate data sets that support a claim or communicate information.
Clarification Statement:	Level 2: Example data collection tools and techniques could include scientific probes, robotics sensors, microcontroller sensors, mobile device applications, etc.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
F.TF.I.5, F.LE.F.1, F.IF.B.5, A.CED.G.1, A.CED.G.2		CV12.5.1, CV12.5.2, CV12.5.4	5b - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH

Domain: Computing Systems

Practice(s): Level 1 & 2: 4.4

By end of Grade 12

Standard: Inference & Models	L1.DA.IM.01 Create computational models that represent the relationships among different elements of data collected from a phenomenon or process.	L2.DA.IM.01 Formulate, refine, and test scientific hypotheses using models and simulations.
Clarification Statement:	Level 1: 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. Students should model phenomena as systems, with rules governing the interactions within the system. Students should analyze and evaluate these models against real world observations. For example, students might create a simple producer–consumer ecosystem model using a programming tool. Eventually, they could progress to creating more complex and realistic interactions between species, such as predation, competition, or symbiosis, and evaluate the model based on data gathered from nature.	

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
L1 —F.TF.I.5, F.LE.F.1, F.IF.B.5, A.CED.G.1, A.CED.G.2, A.CED.G.3, S.IC.D.2, S.IC.E.6, S.ID.B.6a L2 —F.TF.I.5, F.LE.F.1, F.IF.B.5, A.CED.G.1, A.CED.G.2, A.CED.G.3, A.CED.G.4	L1 —HS-ETS1-1, HS-ESS3-5, HS-ESS3-6, HS-ETS1-1, HS-ETS1-4	L1 —CV12.2.1, CV12.3.1, CV12.5.2, CV12.5.4 L2 —CV12.5.2	L1 —5b - Computational Thinker L2 —4c - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
	L1 —SS12.5.1, SS12.5.1.a		

Domain: Algorithms & Programming
Practice(s): Level 1: 5.2; Level 2: 4.2
By end of Grade 12

Standard: Algorithms	L1.AP.A.01 Create a prototype that uses algorithms (e. g., searching, sorting, finding shortest distance) to provide a possible solution for a real-world problem relevant to the student.	L2.AP.A.01 Critically examine and trace classic algorithms. Use and adapt classic algorithms to solve computational problems (e.g., selection sort, insertion sort, binary search, linear search).
Clarification Statement:	Level 1: The process of developing computational artifacts embraces both creative expression and the exploration of ideas to create prototypes and solve computational problems. Students create artifacts that are personally relevant or beneficial to their community and beyond. Students should develop artifacts in response to a task or a computational problem that demonstrate the performance, reusability, and ease of implementation of an algorithm. 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.	

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
L1—F.IF.A.1 L2—F.IF.A.1, F.IF.A.3, F.IF.C.9		L1—CV12.3.1, CV12.4.4, CV12.5.1, CV12.5.2, CV12.5.4 L2—CV12.4.4, CV12.5.1, CV12.5.2, CV12.5.4	L1—4a, 4d - Innovative Designer L2—4a - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Algorithms & Programming

Practice(s): Level 1: 7.2; Level 2: 5.2, 5.3

By end of Grade 12

Standard: Algorithms	L1.AP.A.02 Describe how artificial intelligence algorithms drive many software and physical systems.	L2.AP.A.02 Develop an artificial intelligence algorithm to play a game against a human opponent or solve a real-world problem.
Clarification Statement:	Level 1: Examples include digital ad delivery, self-driving cars, computer vision, text analysis, autonomous robots, pattern recognition, and credit card fraud detection.	Level 2: Games do not have to be complex. Simple guessing games, Tic-Tac-Toe, or simple robot commands would be sufficient.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
L2—F.BF.D.1		L1—CV12.2.1, CV12.3.3 L2—CV12.3.1, CV12.5.1, CV12.5.2, CV12.5.4	L1—5d - Computational Thinker L2—4d - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			L2—HE12.4.10



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Algorithms & Programming

Practice(s): 4.2

By end of Grade 12

Standard:
Algorithms

L2.AP.A.03 Evaluate algorithms (e.g., sorting, searching) in terms of their efficiency, correctness, and clarity.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
F.IF.C.9, F.IF.B.4, F.LE.F.5		CV12.2.1, CV12.3.3	5a - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH

Domain: Algorithms & Programming
Practice(s): Level 1: 4.1; Level 2: 4.2

By end of Grade 12

Standard: Variables	L1.AP.V.01 Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.	L2.AP.V.01 Compare and contrast simple data structures and their uses (e.g., lists, stacks, queues).
Clarification Statement:	Level 1: Students should be able to identify common features in multiple segments of code and substitute a single segment that uses lists (or arrays) to account for the differences.	Level 2: Examples could include lists, arrays, stacks, and queues.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L1—CV12.5.1 L2—CV12.2.1, CV12.3.3	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Algorithms & Programming

Practice(s): 5.2

By end of Grade 12

Standard: Control	L1.AP.C.01 Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.
Clarification Statement:	Level 1: Implementation includes the choice of programming language, which affects the time and effort required to create a program. Readability refers to how clear the program is to other programmers and can be improved through documentation. The discussion of performance is limited to a theoretical understanding of execution time and storage requirements; a quantitative analysis is not expected. Control structures at this level may include conditional statements, loops, event handlers, and recursion. For example, students might compare the readability and program performance of iterative and recursive implementations of procedures that calculate the Fibonacci sequence. Students may also consider the effects of caching to improve performance.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.2.1, CV12.3.3	4a - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Algorithms & Programming

Practice(s): Level 1 & 2: 3.2

By end of Grade 12

Standard: Control	L1.AP.C.02 Trace the execution of loops and conditional statements, illustrating output and changes in values of named variables.	L2.AP.C.01 Trace the execution of recursion, illustrating output and changes in values of named variables.
Clarification Statement:	Level 1: For example, tracing a for loop could include the value of variables and how they change each time through the loop.	Level 2: The trace could include the input arguments and the return values of each recursive call. Nesting according to recursive invocation may be used to organize the trace.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
L1 & L2—F.IF.A.1, F.IF.A.3			L1 & L2—4a - Innovative Designer L1 & L2—5c - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Algorithms & Programming

Practice(s): 5.2

By end of Grade 12

Standard: Control	L1.AP.C.03 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.
Clarification Statement:	Level 1: 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. In L1.AP.M.01, students learn to create and call procedures. In this standard, students design procedures that are called by events. Students might create a mobile app that updates a list of nearby points of interest when the device detects that its location has been changed.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.3.4, CV12.5.1, CV12.5.2, CV12.5.4	3d - Knowledge Constructor 4c - Innovative Designer 5a - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Algorithms & Programming

Practice(s): Level 1: 3.2; Level 2: 4.3, 5.2

By end of Grade 12

Standard: Modularity	L1.AP.M.01 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.	L2.AP.M.01 Construct solutions to problems using student-created components, such as procedures, modules, and/or objects.
Clarification Statement:	Level 1: At this level, students should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist. For example, students could create an app to solve a community problem by connecting to an online database through an application programming interface (API).	Level 2: Object-oriented programming is optional at this level. Problems can be assigned or student-selected.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	L1—HS-ETS1-2	L2—CV12.3.1, CV12.5.1, CV12.5.2, CV12.5.4	L1—5c - Computational Thinker L2—3d - Knowledge Constructor
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH

Domain: Algorithms & Programming
Practice(s): Level 1: 5.2; Level 2: 4.1
By end of Grade 12

Standard: Modularity	L1.AP.M.02 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.	L2.AP.M.02 Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution.
Clarification Statement:	Level 1: Computational artifacts can be created by combining and modifying existing artifacts or by developing new artifacts. Examples of computational artifacts could include programs, simulations, visualizations, digital animations, robotic systems, and apps. Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. Modules allow for better management of complex tasks. The focus at this level is understanding a program as a system with relationships between modules. The choice of implementation, such as programming language or paradigm, may vary. For example, 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.	Level 2: As students encounter complex, real-world problems that span multiple disciplines or social systems, they should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist. This standard is similar to L1.AP.M.01. The difference is that this standard expects greater complexity in the real-world problem that is being solved.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L1 —CV12.4.4, CV12.5.1, CV12.5.2, CV12.5.4 L2 —CV12.3.2, CV12.5.2	L1 —4a - Innovative Designer L2 —3d - Knowledge Constructor
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Algorithms & Programming

Practice(s): 4.2, 5.3

By end of Grade 12

Standard: Modularity	L2.AP.M.03 Demonstrate code reuse by creating programming solutions using libraries and APIs.
Clarification Statement:	Level 2: Libraries and APIs can be student-created or common graphics libraries or map APIs, for example.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.5.1, CV12.5.2, CV12.5.4	5c - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Algorithms & Programming

Practice(s): Level 1 & 2: 5.1

By end of Grade 12

Standard:	L1.AP.PD.01 Plan and develop programs by analyzing a problem and/or process, developing and documenting a solution, testing outcomes, and adapting the program for a variety of users.	L2.AP.PD.01 Plan and develop programs that will provide solutions to a variety of users using a software life cycle process.
Program Development		
Clarification Statement:		Level 2: Processes could include agile, spiral, or waterfall.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L1—CV12.5.1, CV12.5.2 L2—CV12.3.1, CV12.5.1, CV12.5.2, CV12.5.4	L1—4c - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
		L1—FPA11.1.A.3, FPA11.1.T.2, FPA11.1.D.7	



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Algorithms & Programming

Practice(s): Level 1: 7.3; Level 2: 2.4

By end of Grade 12

Standard:	L1.AP.PD.02 Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.	L2.AP.PD.02 Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (e.g., code documentation) in a group software project.
Program Development		
Clarification Statement:	Level 1: Examples of software licenses could include commercial, freeware, and the many open-source licensing schemes. 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.	Level 2: Processes could include agile, spiral, or waterfall.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L2—CV12.2.3, CV12.5.2	L1—2c - Digital Citizen L2—7b - Global Collaborator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
		L2—FL2.IL.1, FL2.IL.2, FL3.IL.1, FL3.IL.2	



Domain: Algorithms & Programming

Practice(s): 6.2

By end of Grade 12

**Standard:
Program
Development**

L1.AP.PD.03 Use debugging tools to identify and fix errors in a program.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.5.2	4c - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH

Domain: Algorithms & Programming
Practice(s): Level 1: 2.4; Level 2: 5.2
By end of Grade 12

Standard: Program Development	L1.AP.PD.04 Design and develop computational artifacts, working in team roles, using collaborative tools.	L2.AP.PD.03 Develop programs for multiple computing platforms.
Clarification Statement:	Level 1: 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. 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.	Level 2: Example platforms could include computer desktop, arduino, robotics, web, or mobile. A student could develop a single program that works on multiple platforms or develop multiple programs that each work on a different platform.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L1—CV12.2.3, CV12.5.2 L2—CV12.5.1, CV12.5.2, CV12.5.4	L1—7b - Global Collaborator L2—4a - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
		L1—FPA11.1.A.4, FPA11.1.T.4, FPA11.1.D.5	



Domain: Algorithms & Programming

Practice(s): Level 1: 7.2; Level 2: 6.3

By end of Grade 12

Standard: Program Development	L1.AP.PD.05 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.	L2.AP.PD.04 Evaluate key qualities of a program through a process such as a code review (e.g., qualities could include correctness, usability, readability, efficiency, portability, and scalability).
Clarification Statement:	Level 1: Creating a program requires making many decisions about modules, roles, communication, control, etc. These decisions are easy to forget, so it is essential that they be documented for future use. Students may use any tools for documentation, including generic word processors, comments within the program, or specialized tools such as Github Wiki.	

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L1—CV12.5.1, CV12.5.2 L2—CV12.5.2	L1—6c - Creative Communicator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
L1—W.9-10.6			



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Algorithms & Programming

Practice(s): Level 1: 6.3; Level 2: 6.1

By end of Grade 12

Standard: Program Development	L1.AP.PD.06 Evaluate and refine computational artifacts to make them more usable and accessible.	L2.AP.PD.05 Develop and use a series of test cases to verify that a program performs according to its design specifications.
Clarification Statement:	Level 1: 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 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.	Level 2: At this level, students are expected to select their own test cases.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L1—CV12.2.1, CV12.5.2 L2—CV12.5.2	L1 & L2—4c - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



Domain: Algorithms & Programming

Practice(s): 7.2

By end of Grade 12

Standard: Program Development	L2.AP.PD.06 Explain security issues that might lead to compromised computer programs.
Clarification Statement:	Level 2: For example, common issues include lack of bounds checking, poor input validation, and circular references.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.2.1	2d - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Algorithms & Programming

Practice(s): 5.3

By end of Grade 12

Standard: Program Development	L2.AP.PD.07 Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
Clarification Statement:	Level 2: For instance, changes made to a method or function signature could break invocations of that method elsewhere in a system.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.5.1, CV12.5.2, CV12.5.4	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Algorithms & Programming

Practice(s): 7.2

By end of Grade 12

Standard: Program Development	L2.AP.PD.08 Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems.
Clarification Statement:	Level 2: Examples of features include blocks versus text, indentation versus curly braces, and high-level versus low-level.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.2.1, CV12.3.1, CV12.3.3	
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Impacts of Computing

Practice(s): Level 1 & 2: 1.2

By end of Grade 12

Standard: Culture	L1.IC.C.01 Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.	L2.IC.C.01 Evaluate the beneficial and harmful effects that computational artifacts and innovations have on society.
Clarification Statement:	Level 1: Computing may improve, harm, or maintain practices. Equity deficits, such as minimal exposure to computing, access to education, and training opportunities, are related to larger, systemic problems in society. Students should be able to evaluate the accessibility of a product to a broad group of end users, such as people who lack access to broadband or who have various disabilities. Students should also begin to identify potential bias during the design process to maximize accessibility in product design.	

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
	L1—HS-PS3-1, HS-ESS3-3 L2—HS-ETS1-3	L1—CV12.2.1, CV12.3.4 L2—CV12.2.1, CV12.3.4, CV12.3.3	L2—2c - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
L1—W.9-10.6		L2—FPA11.1.A.5, FPA11.4.A.4, FPA11.4.M.1, FPA11.4.T.2, FPA11.4.D.3, FPA11.3.D.3	



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Impacts of Computing

Practice(s): Level 1 & 2: 1.2

By end of Grade 12

Standard: Culture	L1.IC.C.02 Test and refine computational artifacts to reduce bias and equity deficits.	L2.IC.C.02 Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society.
Clarification Statement:	Level 1: Biases could include incorrect assumptions developers have made about their user base. Equity deficits include minimal exposure to computing, access to education, and training opportunities. Students should begin to identify potential bias during the design process to maximize accessibility in product design and become aware of professionally accepted accessibility standards to evaluate computational artifacts for accessibility.	

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L1—CV12.5.1, CV12.5.2 L2—CV12.2.1, CV12.3.4, CV12.5.2	L1—4c - Innovative Designer
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			L2—HE12.4.10



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Impacts of Computing

Practice(s): Level 1: 3.1; Level 2: 5.2

By end of Grade 12

Standard: Culture	L1.IC.C.03 Demonstrate how a given algorithm applies to problems across disciplines.	L2.IC.C.03 Predict how computational innovations that have revolutionized aspects of our culture might evolve.
Clarification Statement:	Level 1: Computation can share features with disciplines such as art and music by algorithmically translating human intention into an artifact. Students should be able to identify real-world problems that span multiple disciplines, such as increasing bike safety with new helmet technology, and that can be solved computationally.	Level 2: Areas to consider might include education, healthcare, art, entertainment, and energy.

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
L1—F.LE.F.1, F.IF.A.2, F.IF.B.5	L1—HS-ETS1-4	L2—CV12.2.1	L1—5a - Computational Thinker
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Impacts of Computing

Practice(s): 2.4

By end of Grade 12

Standard: Social Interactions	L1.IC.SI.01 Use tools and methods for collaboration.
Clarification Statement:	Level 1: Many aspects of society, especially careers, have been affected by the degree of communication afforded by computing. The increased connectivity between people in different cultures and in different career fields has changed the nature and content of many careers. Students should explore different collaborative tools and methods used to solicit input from team members, classmates, and others, such as participation in online forums or local communities. For example, students could compare ways different social media tools could help a team become more cohesive.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.2.3	7b - Global Collaborator
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
	SS12.6.3	FPA11.1.A.4, FPA11.1.T.4	



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Impacts of Computing

Practice(s): Level 1 & 2: 1.1, 7.3

By end of Grade 12

Standard: Social Interactions	L1.IC.SI.02 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.	L2.IC.SI.01 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.
--	---	---

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L1 & L2—CV12.2.2, CV12.2.3, CV12.2.4, CV12.5.3	L1 & L2—2b - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
		L1 & L2—FPA11.1.A.5, FPA11.4.A.4, FPA11.4.M.1, FPA11.4.T.2	L1 & L2—HE12.4.11



Domain: Algorithms & Programming

Practice(s): Level 1: 7.3; Level 2: 3.3, 7.3

By end of Grade 12

Standard: Safety, Law, & Ethics	L1.IC.SLE.01 Explain the beneficial and harmful effects that intellectual property laws can have on innovation.	L2.IC.SLE.01 Debate laws and regulations that impact the development and use of software and technology.
Clarification Statement:	Level 1: Laws govern many aspects of computing, such as privacy, data, property, information, and identity. These laws can have beneficial and harmful effects, such as expediting or delaying advancements in computing and protecting or infringing upon people’s rights. International differences in laws and ethics have implications for computing. For example, laws that mandate the blocking of some file-sharing websites may reduce online piracy but can restrict the right to access information. Firewalls can be used to block harmful viruses and malware but can also be used for media censorship. Students should be aware of intellectual property laws and be able to explain how they are used to protect the interests of innovators and how patent trolls abuse the laws for financial gain.	

Wyoming Cross-Disciplinary Connections & ISTE Standards

2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L1 —CV12.2.1, CV12.3.3 L2 —CV12.2.1, CV12.3.3, CV12.5.3	L1 & L2 —2c - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



Domain: Impacts of Computing

Practice(s): 7.2

By end of Grade 12

Standard: Safety, Law, & Ethics	L1.IC.SLE.02 Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.
Clarification Statement:	Level 1: 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 covert 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.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.2.1, CV12.3.3	2d - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Impacts of Computing

Practice(s): 7.3

By end of Grade 12

Standard: Safety, Law, & Ethics	L1.IC.SLE.03 Evaluate the social and economic implications of privacy in the context of safety, law, or ethics.
Clarification Statement:	Level 1: Laws govern many aspects of computing, such as privacy, data, property, information, and identity. International differences in laws and ethics have implications for computing. Students might review case studies or current events which present an ethical dilemma when an individual's right to privacy is at odds with the safety, security, or wellbeing of a community.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		CV12.2.1, CV12.3.3	2b - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			HE12.4.10



2019 Wyoming Computer Science Standards

Grade Band: 9-12

Domain: Algorithms & Programming

Practice(s): Level 1 & 2: 7.2

By end of Grade 12

Standard: Safety, Law, & Ethics	L1.IC.SLE.04 Using grade level appropriate content and complexity, discuss the legal, social, and ethical impacts associated with software development and use, including both positive and malicious intent.	L2.IC.SLE.02 Using grade level appropriate content and complexity, discuss the legal, social, and ethical impacts associated with software development and use, including both positive and malicious intent.
Clarification Statement:	Level 1: Examples of positive impacts could include writing software or utilities to improve communication for people who have a disability, writing an application that manages money for a bank, or software that handles healthcare records. Examples of negative impacts could include distributing a virus, or writing backdoor code, malware, or ransomware.	Level 2: Examples of positive impacts could include writing software or utilities to improve communication for people who have a disability, writing an application that manages money for a bank, or software that handles healthcare records. Examples of negative impacts could include distributing a virus, or writing backdoor code, malware, or ransomware.

Wyoming Cross-Disciplinary Connections & ISTE Standards			
2018 MATH	2016 SCIENCE	2014 C&VE	2016 ISTE / WY DL GUIDELINES
		L1 & L2—CV12.2.1, CV12.2.2, CV12.2.4	L1 & L2—2a - Digital Citizen
2012 ELA	2018 SOCIAL STUDIES	2013 FINE & PERFORMING ARTS	2014 P.E. / 2012 HEALTH
			L1 & L2—HE12.4.10

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Devices: L1.CS.D.01 Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.	provides little to no evidence in addressing the expectation(s).	identifies abstractions that hide the underlying implementation details of computing systems embedded in everyday objects.	explains how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Hardware & Software: L1.CS.HS.01 Explain the interactions between application software, system software, and hardware layers.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - identifies application software, system software, and hardware layers. - defines application software, system software, and hardware layers. 	<ul style="list-style-type: none"> - identifies the interactions between application software, system software, and hardware layers. - defines the interactions between application software, system software, and hardware layers. - explains the interactions between application software, system software, and hardware layers. e.g., 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. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., student demonstrates knowledge of specific, advanced terms for computer architecture, such as BIOS, kernel, or bus).
Hardware & Software: L2.CS.HS.01 Categorize the roles of operating system software.	provides little to no evidence in addressing the expectation(s).	categorizes some of the roles of operating system software.	categorizes the roles of the operating system software (e.g., roles could include memory management, data storage/retrieval, process management, and access control).	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Troubleshooting: L1.CS.T.01 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and resolve errors.	provides little to no evidence in addressing the expectation(s).	develops guidelines with support that convey systematic troubleshooting strategies that others can use to identify and resolve errors.	develops guidelines independently that convey systematic troubleshooting strategies that others can use to identify and resolve errors (e.g., students could create a flow chart, a job aid for a help desk employee, or an expert system).	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., someone with limited experience or knowledge could follow student developed guidelines).
Troubleshooting: L2.CS.T.01 Identify how hardware components facilitate logic, input, output, and storage in computing systems, and their common malfunctions.	provides little to no evidence in addressing the expectation(s).	identifies how some hardware components: - facilitate logic, input, output, and storage in computing systems, and/or - some of their common malfunctions.	identifies: - how hardware components facilitate logic, input, output, and storage in computing systems. - hardware components common malfunctions.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Network Communication & Organization: L1.NI.NCO.01 Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - identifies routers, switches, servers, topology, and addressing. - defines routers, switches, servers, topology, and addressing. 	by describing the relationship between routers, switches, servers, topology, and addressing, evaluates: <ul style="list-style-type: none"> - the scalability of networks. - the reliability of networks. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., students can discuss different types of routers, switches, servers and/or topologies).
Network Communication & Organization: L2.NI.NCO.01 Describe the issues that impact network functionality (e.g., bandwidth, load, latency, topology).	provides little to no evidence in addressing the expectation(s).	describes a limited number of issues that impact network functionality (e.g., bandwidth, load, latency, topology).	describes common issues that impact network functionality (e.g., bandwidth, load, latency, topology).	demonstrates an understanding of trade-offs between network functionality and design.
Cybersecurity: L1.NI.C.01 Give examples to illustrate how sensitive data can be affected by malware and other attacks.	provides little to no evidence in addressing the expectation(s).	recalls examples to illustrate how sensitive data can be affected by malware and other attacks.	gives multiple detailed examples to illustrate how sensitive data can be affected by malware and other attacks.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Cybersecurity: L2.NI.C.01 Compare ways software developers protect devices and information from unauthorized access.	provides little to no evidence in addressing the expectation(s).	lists ways software developers protect: <ul style="list-style-type: none"> - devices from unauthorized access. - information from unauthorized access. 	compares ways software developers protect: <ul style="list-style-type: none"> - devices from unauthorized access. - information from unauthorized access. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., encryption strategies, authentication strategies).

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Cybersecurity: L1.NI.C.02 Recommend cybersecurity measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.	provides little to no evidence in addressing the expectation(s).	identifies cybersecurity measures to address various scenarios.	recommends cybersecurity measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Cybersecurity: L1.NI.C.03 Compare various security measures, considering trade-offs between the usability and security of a computing system.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - identifies various security measures. - defines various security measures. 	compares various security measures, considering trade-offs between the usability and security of a computing system.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., discuss security policies that are in place that present a trade-off between usability and security).
Cybersecurity: L1.NI.C.04 Explain trade-offs when selecting and implementing cybersecurity recommendations.	provides little to no evidence in addressing the expectation(s).	when selecting and implementing cybersecurity recommendations, can give an example of trade-offs: <ul style="list-style-type: none"> - from a single viewpoint, and/or <ul style="list-style-type: none"> - with inappropriate terminology. 	explains trade-offs from multiple perspectives using appropriate terminology when selecting and implementing cybersecurity recommendations.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., make a recommendation and justify).

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Storage: L1.DA.S.01 Translate between different bit representations of real-world phenomena, such as characters, numbers, and images.	provides little to no evidence in addressing the expectation(s).	can translate between a bit representation of real-world phenomena, such as characters, numbers, or images.	translates between different bit representations of real-world phenomena, such as characters, numbers, and images.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Storage: L1.DA.S.02 Evaluate the trade-offs in how data elements are organized and where data is stored.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - identifies the trade-offs in how data elements are organized and where data is stored. - describes the trade-offs in how data elements are organized and where data is stored. 	evaluates the trade-offs in how data elements are organized and where data is stored.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., research emerging technologies for data storage and evaluate trade-off with current technologies).

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Collection, Visualization, & Transformation: L1.DA.CVT.01 Create interactive data representations using software tools to help others better understand real-world phenomena (e.g., paper surveys and online data sets).	provides little to no evidence in addressing the expectation(s).	creates, with errors, interactive data representations using software tools.	creates, with no or minor errors, appropriate interactive data representations using software tools to help others better understand real-world phenomena.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., research emerging visualization techniques and use them to create new data representations).
Collection, Visualization, & Transformation: L2.DA.CVT.01 Use data analysis tools and techniques to identify patterns in data representing complex systems.	provides little to no evidence in addressing the expectation(s).	uses data analysis tools and techniques to identify patterns in data representing complex systems but draws incorrect conclusions.	uses data analysis tools and techniques to identify correct patterns in data representing complex systems.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., make a plausible predication based on pattern).
Collection, Visualization, & Transformation: L2.DA.CVT.02 Select data collection tools and techniques, and use them to generate data sets that support a claim or communicate information.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - selects data collection tools and techniques. - uses data collection tools and techniques to generate data sets but are unable to support a claim or communicate information. 	<ul style="list-style-type: none"> - selects data collection tools and techniques. - uses data collection tools to generate data sets that support a claim or communicate information. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Inference & Models: L1.DA.IM.01 Create computational models that represent the relationships among different elements of data collected from a phenomenon or process.	provides little to no evidence in addressing the expectation(s).	creates computational models that represent the relationships among different elements of data collected from a phenomenon or process.	creates accurate computational models that represent the relationships among different elements of data collected from a phenomenon or process.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Inference & Models: L2.DA.IM.01 Formulate, refine, and test scientific hypotheses using models and simulations.	provides little to no evidence in addressing the expectation(s).	formulates scientific hypotheses using models and simulations.	<ul style="list-style-type: none"> - formulates scientific hypotheses using models and simulations. - refines scientific hypotheses using models and simulations. - tests scientific hypotheses using models and simulations. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Algorithms: L1.AP.A.01 Create a prototype that uses algorithms (e. g., searching, sorting, finding shortest distance) to provide a possible solution for a real-world problem relevant to the student.	provides little to no evidence in addressing the expectation(s).	creates a prototype that uses an algorithm (e. g., searching, sorting, finding shortest distance) to provide a possible solution for a real-world problem relevant to the student.	creates a prototype that uses multiple algorithms (e. g., searching, sorting, finding shortest distance) to provide a possible solution for a real-world problem relevant to the student.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., student generated problem).
Algorithms: L2.AP.A.01 Critically examine and trace classic algorithms. Use and adapt classic algorithms to solve computational problems (e.g., selection sort, insertion sort, binary search, linear search).	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - examines and traces classic algorithms with minor errors. - uses classic algorithms to solve computational problems. 	<ul style="list-style-type: none"> - critically examines and traces classic algorithms. - uses classic algorithms to solve computational problems. - adapts classic algorithms to solve computational problems. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., use and justify why a given algorithm is more efficient than another).

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Algorithms: L1.AP.A.02 Describe how artificial intelligence algorithms drive many software and physical systems.	provides little to no evidence in addressing the expectation(s).	describes how artificial intelligence algorithms drive a software system or physical system.	describes how artificial intelligence algorithms drive many software and physical systems.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., student discusses different types of artificial intelligence algorithms).
Algorithms: L2.AP.A.02 Develop an artificial intelligence algorithm to play a game against a human opponent or solve a real-world problem.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - develops an artificial intelligence algorithm to play a game against a human opponent or solve a real-world problem. - incorrectly captures some rules of the game. 	<ul style="list-style-type: none"> - develops an artificial intelligence algorithm to play a game against a human opponent or solve a real-world problem. - correctly implements all rules of the game. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., uses heuristics to select the moves of the computer).
Algorithms: L2.AP.A.03 Evaluate algorithms (e.g., sorting, searching) in terms of their efficiency, correctness, and clarity.	provides little to no evidence in addressing the expectation(s).	evaluates algorithms in terms of their: <ul style="list-style-type: none"> - efficiency, or <ul style="list-style-type: none"> - correctness, or <ul style="list-style-type: none"> - clarity. 	evaluates algorithms in terms of their: <ul style="list-style-type: none"> - efficiency. - correctness. - clarity. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Variables: L1.AP.V.01 Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.	provides little to no evidence in addressing the expectation(s).	with guidance, uses lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.	independently uses lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., uses standard list operations like filter, map, and reduce).
Variables: L2.AP.V.01 Compare and contrast simple data structures and their uses (e.g., lists, stacks, queues).	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - identifies simple linear data structures and their uses. - explains simple linear data structures and their uses. 	compares and contrasts simple linear data structures and their uses.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., trees).

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Control: L1.AP.C.01 Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.	provides little to no evidence in addressing the expectation(s).	justifies the selection of specific control structures when tradeoffs involve: - implementation, or - readability, or - program performance.	- justifies the selection of specific control structures when tradeoffs involve implementation, readability, and program performance. - explains the benefits and drawbacks of choices.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., exception handling).
Control: L1.AP.C.02 Trace the execution of loops and conditional statements, illustrating output and changes in values of named variables.	provides little to no evidence in addressing the expectation(s).	traces the execution of: - loops illustrating output and changes in values of named variables, or - conditional statements illustrating output and changes in values of named variables.	traces the execution of: - loops illustrating output and changes in values of named variables, and - conditional statements illustrating output and changes in values of named variables.	In addition to the proficient level, student demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Control: L2.AP.C.01 Trace the execution of recursion, illustrating output and changes in values of named variables.	provides little to no evidence in addressing the expectation(s).	with guidance: - traces the execution of recursion. - illustrates output and changes in values of name variables.	independently: - traces the execution of linear recursion. - illustrates output and changes in values of name variables (e.g., factorial function).	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., Fibonacci).

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Control: L1.AP.C.03 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.	provides little to no evidence in addressing the expectation(s).	designs computational artifacts that uses events to initiate instructions.	<ul style="list-style-type: none"> - designs computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions. - iteratively develops computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., using multiple user interface components).
Modularity: L1.AP.M.01 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.	provides little to no evidence in addressing the expectation(s).	decomposes problems into smaller components that are incohesive or tightly coupled.	decomposes problems into smaller components that are highly cohesive and loosely coupled through systematic analysis, using constructs such as procedures, modules, and/or objects.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., an appropriate class hierarchy).
Modularity: L2.AP.M.01 Construct solutions to problems using student-created components, such as procedures, modules, and/or objects.	provides little to no evidence in addressing the expectation(s).	with guidance, constructs solutions to problems using student-created components, such as procedures, modules, and/or objects.	constructs solutions to problems using student-created components, such as procedures, modules, and/or objects.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Modularity: L1.AP.M.02 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.	provides little to no evidence in addressing the expectation(s).	with guidance, creates artifacts by using: - procedures within a program, or - combinations of data and procedures, or - independent but interrelated programs.	independently, creates artifacts by using: - procedures within a program, or - combinations of data and procedures, or - independent but interrelated programs.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Modularity: L2.AP.M.02 Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution.	provides little to no evidence in addressing the expectation(s).	- analyzes a large-scale computational problem and with guidance. - identifies generalizable patterns that can be applied to a solution.	- analyzes a large-scale computational problem. - independently identifies generalizable patterns that can be applied to a solution.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Modularity: L2.AP.M.03 Demonstrate code reuse by creating programming solutions using libraries and APIs.	provides little to no evidence in addressing the expectation(s).	with guidance, demonstrates code reuse by creating programming solutions using libraries and APIs.	independently, demonstrates code reuse by creating programming solutions using libraries and APIs.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Program Development: L1.AP.PD.01 Plan and develop programs by analyzing a problem and/or process, developing and documenting a solution, testing outcomes, and adapting the program for a variety of users.	provides little to no evidence in addressing the expectation(s).	with instructor support, plans and develops programs by: <ul style="list-style-type: none"> - analyzing a problem and/or process. - developing and documenting a solution. - testing outcomes. 	plans and develops programs by: <ul style="list-style-type: none"> - analyzing a problem and/or process. - developing and documenting a solution. - testing outcomes. - adapting the program for a variety of users. 	independently plans and develops programs by: <ul style="list-style-type: none"> - analyzing a problem and/or process. - developing and documenting a solution. - testing outcomes. - adapting the program for a variety of users.
Program Development: L2.AP.PD.01 Plan and develop programs that will provide solutions to a variety of users using a software life cycle process.	provides little to no evidence in addressing the expectation(s).	with instructor support: <ul style="list-style-type: none"> - plans a program that will provide solutions to a variety of users using a software life cycle process. - develops a program that will provide solutions to a variety of users using a software life cycle process. 	<ul style="list-style-type: none"> - plans a program that will provide solutions to a variety of users using a software life cycle process. - develops a program that will provide solutions to a variety of users using a software life cycle process. 	independently: <ul style="list-style-type: none"> - plans a program that will provide solutions to a variety of users using a software life cycle process. - develops a program that will provide solutions to a variety of users using a software life cycle process.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Program Development: L1.AP.PD.02 Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - identifies licenses that limit or restrict use of computational artifacts when using resources such as libraries. - defines licenses that limit or restrict use of computational artifacts when using resources such as libraries. 	evaluates licenses that limit or restrict use of computational artifacts when using resources such as libraries (e.g. students might consider two software libraries that address a similar need, justifying their choice based on the library that has the least restrictive license).	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Program Development: L2.AP.PD.02 Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (e.g., code documentation) in a group software project.	provides little to no evidence in addressing the expectation(s).	uses: <ul style="list-style-type: none"> - integrated development environments (IDEs) in a group software project. - collaborative tools or practices (e.g., code documentation) in a group software project. 	uses: <ul style="list-style-type: none"> - version control systems in a group software project. - integrated development environments (IDEs) in a group software project. - collaborative tools and practices (e.g., code documentation) in a group software project. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Program Development: L1.AP.PD.03 Use debugging tools to identify and fix errors in a program.	provides little to no evidence in addressing the expectation(s).	identifies strategies to test and debug (identify and fix errors) a program or algorithm to ensure it runs.	tests and debugs (identify and fix errors) a program or algorithm to ensure it runs as intended.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Program Development: L1.AP.PD.04 Design and develop computational artifacts, working in team roles, using collaborative tools.	provides little to no evidence in addressing the expectation(s).	- designs computational artifacts using collaborative tools. - develops computational artifacts using collaborative tools.	designs and develops computational artifacts, working in team roles, using collaborative tools (e.g., team roles in pair programming are driver and navigator but could be more specialized in larger teams).	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard. As programs grow more complex, the choice of resources that aid program development becomes increasingly important and should be made by the students.
Program Development: L2.AP.PD.03 Develop programs for multiple computing platforms.	provides little to no evidence in addressing the expectation(s).	with instructor support, develops programs for multiple computing platforms.	develops programs for multiple computing platforms (e.g., disparate programs for different platforms: computer desktop, web, or mobile).	develops programs for multiple cross-platform computing platforms (e.g., platforms could include: computer desktop, web, or mobile).
Program Development: L1.AP.PD.05 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.	provides little to no evidence in addressing the expectation(s).	partially documents design decisions using: - text, graphics, presentations, and/or - demonstrations in the development of complex programs.	documents design decisions using: - text, graphics, presentations, and/or - demonstrations in the development of complex programs.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Program Development: L2.AP.PD.04 Evaluate key qualities of a program through a process such as a code review (e.g., qualities could include correctness, usability, readability, efficiency, portability, and scalability).	provides little to no evidence in addressing the expectation(s).	identifies key qualities of a program. - defines key qualities of a program (e.g., correctness, usability, readability, efficiency, portability, and scalability).	evaluates key qualities of a program through a process such as a code review (e.g., correctness, usability, readability, efficiency, portability, and scalability).	evaluates key qualities of a program and makes recommendations to improve that program through a process such as a code review (e.g., correctness, usability, readability, efficiency, portability, and scalability).
Program Development: L1.AP.PD.06 Evaluate and refine computational artifacts to make them more usable and accessible.	provides little to no evidence in addressing the expectation(s).	with support: - evaluates computational artifacts to make them more usable and accessible. - refines computational artifacts to make them more usable and accessible.	- evaluates computational artifacts to make them more usable and accessible. - refines computational artifacts to make them more usable and accessible.	supports others as they: - evaluate computational artifacts to make them more usable and accessible. -refine computational artifacts to make them more usable and accessible.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Program Development: L2.AP.PD.05 Develop and use a series of test cases to verify that a program performs according to its design specifications.	provides little to no evidence in addressing the expectation(s).	uses a series of test cases to verify that a program performs according to its design specifications.	develops a series of test cases to verify that a program performs according to its design specifications. - uses 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.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Program Development: L2.AP.PD.06 Explain security issues that might lead to compromised computer programs.	provides little to no evidence in addressing the expectation(s).	- identifies security issues that might lead to compromised computer programs. - describes security issues that might lead to compromised computer programs.	explains security issues that might lead to compromised computer programs (e.g., lack of bounds checking, poor input validation, and circular references).	explains and provides potential solutions for security issues that might lead to compromised computer programs.
Program Development: L2.AP.PD.07 Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).	provides little to no evidence in addressing the expectation(s).	modifies an existing program to add additional functionality.	- modifies an existing program to add additional functionality. - discusses intended and unintended implications (e.g., breaking other functionality).	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Program Development: L2.AP.PD.08 Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - identifies multiple programming languages. - explains multiple programming languages. 	<ul style="list-style-type: none"> - compares multiple programming languages. - discusses how their features make them suitable for solving different types of problems. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Culture: L1.IC.C.01 Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - identifies the ways computing impacts personal, ethical, social, economic, and cultural practices. - defines the ways computing impacts personal, ethical, social, economic, and cultural practices. 	evaluates the ways computing impacts personal, ethical, social, economic, and cultural practices.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Culture: L2.IC.C.01 Evaluate the beneficial and harmful effects that computational artifacts and innovations have on society.	provides little to no evidence in addressing the expectation(s).	<ul style="list-style-type: none"> - identifies the beneficial and harmful effects that computational artifacts and innovations have on society. - defines the beneficial and harmful effects that computational artifacts and innovations have on society. 	evaluates the beneficial and harmful effects that computational artifacts and innovations have on society.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Culture: L1.IC.C.02 Test and refine computational artifacts to reduce bias and equity deficits.	provides little to no evidence in addressing the expectation(s).	identifies how computational artifacts reduce bias and equity deficits.	<ul style="list-style-type: none"> - tests computational artifacts to reduce bias and equity deficits. - refines computational artifacts to reduce bias and equity deficits. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., student creates a computational artifact that utilizes accepted accessibility standards).
Culture: L2.IC.C.02 Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society.	provides little to no evidence in addressing the expectation(s).	provides examples for how: <ul style="list-style-type: none"> - equity impacts the distribution of computing resources in a global society. - access impacts the distribution of computing resources in a global society. - influence impacts the distribution of computing resources in a global society. 	evaluates the impact of: <ul style="list-style-type: none"> - equity on the distribution of computing resources in a global society. - access on the distribution of computing resources in a global society. - influence on the distribution of computing resources in a global society. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Culture: L1.IC.C.03 Demonstrate how a given algorithm applies to problems across disciplines.	provides little to no evidence in addressing the expectation(s).	identifies several disciplines a given algorithm applies to.	demonstrates how a given algorithm applies to problems across disciplines.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Culture: L2.IC.C.03 Predict how computational innovations that have revolutionized aspects of our culture might evolve.	provides little to no evidence in addressing the expectation(s).	identifies computational innovations that have revolutionized aspects of our culture.	predicts how computational innovations, that have revolutionized aspects of our culture, might evolve.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Social Interactions: L1.IC.SI.01 Use tools and methods for collaboration.	provides little to no evidence in addressing the expectation(s).	uses basic tools and methods for collaboration.	uses a variety of tools and methods for collaboration.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard (e.g., students could compare and recommend ways different tools could help a team become more cohesive).
Social Interactions: L1.IC.SI.02 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.	provides little to no evidence in addressing the expectation(s).	generally practices grade-level appropriate behavior and responsibilities while participating in an online community.	<ul style="list-style-type: none"> - practices grade-level appropriate behavior and responsibilities while participating in an online community. - identifies and reports inappropriate behavior. 	models grade-level appropriate behavior and responsibilities while participating in an online community.
Social Interactions: L2.IC.SI.01 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.	provides little to no evidence in addressing the expectation(s).	generally practices grade-level appropriate behavior and responsibilities while participating in an online community.	<ul style="list-style-type: none"> - practices grade-level appropriate behavior and responsibilities while participating in an online community. - identifies and reports inappropriate behavior. 	models grade-level appropriate behavior and responsibilities while participating in an online community.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Safety, Law, & Ethics: L1.IC.SLE.01 Explain the beneficial and harmful effects that intellectual property laws can have on innovation.	provides little to no evidence in addressing the expectation(s).	- identifies a beneficial effect intellectual property laws have had on innovation, and/or - identifies a harmful effect intellectual property laws have had on innovation.	- identifies a beneficial effect intellectual property laws have had on innovation. - identifies a harmful effect intellectual property laws have had on innovation.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Safety, Law, & Ethics: L2.IC.SLE.01 Debate laws and regulations that impact the development and use of software and technology.	provides little to no evidence in addressing the expectation(s).	- identifies laws and regulations that impact the development and use of software and technology. - defines laws and regulations that impact the development and use of software and technology.	debates laws and regulations that impact the development and use of software and technology.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Safety, Law, & Ethics: L1.IC.SLE.02 Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.	provides little to no evidence in addressing the expectation(s).	identifies the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.	explains the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Safety, Law, & Ethics: L1.IC.SLE.03 Evaluate the social and economic implications of privacy in the context of safety, law, or ethics.	provides little to no evidence in addressing the expectation(s).	provides examples of the: <ul style="list-style-type: none"> - social implications of privacy in the context of safety, law, or ethics. - economic implications of privacy in the context of safety, law, or ethics. 	evaluates the: <ul style="list-style-type: none"> - social implications of privacy in the context of safety, law, or ethics. - economic implications of privacy in the context of safety, law, or ethics. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

Performance Level Descriptors (PLDs)

Grade Band: 9-12

Standard: Benchmark	The Below Basic student:	The Basic student:	The Proficient student:	In addition to the Proficient Level, the Advanced student:
Safety, Law, & Ethics: L1.IC.SLE.04 Using grade level appropriate content and complexity, discuss the legal, social, and ethical impacts associated with software development and use, including both positive and malicious intent.	provides little to no evidence in addressing the expectation(s).	provides examples of the: <ul style="list-style-type: none"> - legal impacts associated with software development and use, including both positive and malicious intent, or - social impacts associated with software development and use, including both positive and malicious intent, or - ethical impacts associated with software development and use, including both positive and malicious intent. 	discusses the: <ul style="list-style-type: none"> - legal impacts associated with software development and use, including both positive and malicious intent. - social impacts associated with software development and use, including both positive and malicious intent. - ethical impacts associated with software development and use, including both positive and malicious intent. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.
Safety, Law, & Ethics: L2.IC.SLE.02 Using grade level appropriate content and complexity, discuss the legal, social, and ethical impacts associated with software development and use, including both positive and malicious intent.	provides little to no evidence in addressing the expectation(s).	provides examples of the: <ul style="list-style-type: none"> - legal impacts associated with software development and use, including both positive and malicious intent, or - social impacts associated with software development and use, including both positive and malicious intent, or - ethical impacts associated with software development and use, including both positive and malicious intent. 	discusses the: <ul style="list-style-type: none"> - legal impacts associated with software development and use, including both positive and malicious intent. - social impacts associated with software development and use, including both positive and malicious intent. - ethical impacts associated with software development and use, including both positive and malicious intent. 	demonstrates in-depth inferences and applications that go beyond the understanding or context of the standard.

APPENDIX A: GLOSSARY for COMPUTER SCIENCE STANDARDS—page 1 of 5

Abstraction (*Process*): The process of reducing complexity by focusing on the main idea. By hiding details irrelevant to the question at hand and bringing together related and useful details, abstraction reduces complexity and allows one to focus on the problem. (*Product*): A new representation of a thing, a system, or a problem that helpfully reframes a problem by hiding details irrelevant to the question at hand. [MDESE, 2016]

Accessibility The design of products, devices, services, or environments for people who experience disabilities. Accessibility standards that are generally accepted by professional groups include the Web Content Accessibility Guidelines (WCAG) 2.0 and Accessible Rich Internet Applications (ARIA) standards. [Wikipedia]

Algorithm A step-by-step process to complete a task.

Analog The defining characteristic of data that is represented in a continuous, physical way. Whereas digital data is a set of individual symbols, analog data is stored in physical media, such as the surface grooves on a vinyl record, the magnetic tape of a VCR cassette, or other non digital media. [Techopedia]

App A type of application software designed to run on a mobile device, such as a smartphone or tablet computer. Also known as a mobile application. [Techopedia]

Artifact Anything created by a human. See computational artifact for the definition used in computer science.

Application Programming Interface (API) A set of subroutine definitions, communication protocols, and tools for building software. [Wikipedia]

Audience Expected end users of a computational artifact or system.

Authentication (verb): The verification of the identity of a person or process. [FOLDOC]

Authentication Factor(s) (noun): may include password, face recognition, fingerprints, PIN numbers, biometrics, smartcard, Virtual Private Networking (VPN) and Remote Access Services (RAS), etc.

Automate To link disparate systems and software so that they become self-acting or self-regulating. [Ross, 2016]

Automation The process of automating.

Boolean A type of data or expression with two possible values: true and false. [FOLDOC]

Bug An error in a software program. It may cause a program to unexpectedly quit or behave in an unintended manner. [Tech Terms] The process of finding and correcting errors (bugs) is called debugging. [Wikipedia]

Code Any set of instructions expressed in a programming language. [MDESE, 2016]

Comment A programmer-readable annotation in the code of a computer program added to make the code easier to understand. Comments are generally ignored by machines. [Wikipedia]

Complexity The minimum amount of resources, such as memory, time, or messages, needed to solve a problem or execute an algorithm. [NIST/DADS]

Component An element of a larger group. Usually, a component provides a particular service or group of related services. [Tech Terms, TechTarget]

Computational Relating to computers or computing methods.

Computational Artifact Anything created by a human using a computational thinking process and a computing device. A computational artifact can be, but is not limited to, a program, image, audio, video, presentation, or web page file. [College Board, 2016]

APPENDIX A: GLOSSARY for COMPUTER SCIENCE STANDARDS—page 2 of 5

Computational Thinking The thought processes involved in formulating a problem and expressing its solutions in such a way that a computer (human or machine) can effectively carry them out.

Computer A machine or device that performs processes, calculations, and operations based on instructions provided by a software or hardware program. [Techopedia]

Computer Science The study of computing principles, design, and applications (hardware & software); the creation, access, and use of information through algorithms and problem solving, and the impact of computing on society.

Computing Any goal-oriented activity requiring, benefiting from, or creating algorithmic processes. [MDESE, 2016]

Computing Device A physical device that uses hardware and software to receive, process, and output information. Computers, mobile phones, and computer chips inside appliances are all examples of computing devices. [CSTA, 2016]

Computing System A collection of one or more computers or computing devices, together with their hardware and software, integrated for the purpose of accomplishing shared tasks. Although a computing system can be limited to a single computer or computing device, it more commonly refers to a collection of multiple connected computers, computing devices, and hardware. [CSTA, 2016]

Conditional A feature of a programming language that performs different computations or actions depending on whether a programmer-specified Boolean condition evaluates to true or false. [MDESE, 2016] (A conditional could refer to a conditional statement, conditional expression, or conditional construct.)

Configuration ([process](#)): Defining the options that are provided when installing or modifying hardware and software or the process of creating the configuration (product). [TechTarget] ([product](#)): The specific hardware and software details that tell exactly what the system is made up of, especially in terms of devices attached, capacity, or capability. [TechTarget]

Connection A physical or wireless attachment between multiple computing systems, computers, or computing devices. [CSTA]

Connectivity A program's or device's ability to link with other programs and devices. [Webopedia]

Control ([in general](#)) The power to direct the course of actions. ([in programming](#)) The use of elements of programming code to direct which actions take place and the order in which they take place. [CSTA, 2016]

Control Structure A programming (code) structure that implements control. Conditionals and loops are examples of control structures. [CSTA, 2016]

Culture A human institution manifested in the learned behavior of people, including their specific belief systems, language(s), social relations, technologies, institutions, organizations, and systems for using and developing resources. [NCSS, 2013]

Cultural Practices The displays and behaviors of a culture.

Cybersecurity The protection against access to, or alteration of, computing resources through the use of technology, processes, and training. [TechTarget]

Data Information that is collected and used for reference or analysis. Data can be digital or nondigital and can be in many forms, including numbers, text, show of hands, images, sounds, or video. [CAS, 2013; Tech Terms]

Data Structure A particular way to store and organize data within a computer program to suit a specific purpose so that it can be accessed and worked with in appropriate ways. [TechTarget]

APPENDIX A: GLOSSARY for COMPUTER SCIENCE STANDARDS—page 3 of 5

Data Type A classification of data that is distinguished by its attributes and the types of operations that can be performed on it. Some common data types are integer, string, Boolean (true or false), and floating-point. [CSTA, 2016]

Debugging The process of finding and correcting errors (bugs) in programs. [MDESE, 2016]

Decompose To break down into components. [MDESE, 2016]

Decomposition Breaking down a problem or system into components. [MDESE, 2016]

Device A unit of physical hardware that provides one or more computing functions within a computing system. It can provide input to the computer, accept output, or both. [Techopedia]

Document / Documentation written text or illustration that accompanies computer software or is embedded in the source code. It either explains how it operates or how to use it, and may mean different things to people in different roles [Wikipedia]

Digital A characteristic of electronic technology that uses discrete values, generally 0 and 1, to generate, store, and process data. [Techopedia]

Digital Citizenship The norms of appropriate, responsible behavior with regard to the use of technology. [MDESE, 2016]

Efficiency A measure of the amount of resources an algorithm uses to find an answer. It is usually expressed in terms of the theoretical computations, the memory used, the number of messages passed, the number of disk accesses, etc. [NIST/DADS]

Encapsulation The technique of combining data and the procedures that act on it to create a type. [FOLDOC]

Encryption The conversion of electronic data into another form, called ciphertext, which cannot be easily understood by anyone except authorized parties. [TechTarget]

End User (or User) A person for whom a hardware or software product is designed (as distinguished from the developers). [TechTarget]

Event Any identifiable occurrence that has significance for system hardware or software. User-generated events include keystrokes and mouse clicks; system-generated events include program loading and errors. [TechTarget]

Event Handler A procedure that specifies what should happen when a specific event occurs. [CSTA, 2016]

Execute To carry out (or “run”) an instruction or set of instructions (program, app, etc.). [FOLDOC]

Execution The process of executing an instruction or set of instructions. [FOLDOC]

Hardware The physical components that make up a computing system, computer, or computing device. [MDESE, 2016]

Hierarchy An organizational structure in which items are ranked according to levels of importance. [TechTarget]

Human-Computer Interaction (HCI) The study of how people interact with computers and to what extent computing systems are or are not developed for successful interaction with human beings. [TechTarget]

Identifier The user-defined, unique name of a program element (such as a variable or procedure) in code. An identifier name should indicate the meaning and usage of the element being named. [Techopedia]

Implementation The process of expressing the design of a solution in a programming language (code) that can be made to run on a computing device.

Inference A conclusion reached on the basis of evidence and reasoning. [Oxford]

APPENDIX A: GLOSSARY for COMPUTER SCIENCE STANDARDS—page 4 of 5

Input (verb): The signals or instructions sent to a computer. [Techopedia]; (noun): A device or component that allows information to be given to a computer [code.org]

Integrity The overall completeness, accuracy, and consistency of data. [Techopedia]

Internet The global collection of computer networks and their connections, all using shared protocols to communicate. [CAS, 2013]

Interactive Involving the repeating of a process with the aim of approaching a desired goal, target, or result. [MDESE, 2016]

Loop A programming structure that repeats a sequence of instructions as long as a specific condition is true. [Tech Terms]

Memory Temporary storage used by computing devices. [MDESE, 2016]

Model A representation of some part of a problem or a system. [MDESE, 2016] **Note:** This definition differs from that used in science.

Modularity The characteristic of a software/web application that has been divided (decomposed) into smaller modules. An application might have several procedures that are called from inside its main procedure. Existing procedures could be reused by recombining them in a new application. [Techopedia]

Module A software component or part of a program that contains one or more procedures. One or more independently developed modules make up a program. [Techopedia]

Network A group of computing devices (personal computers, phones, servers, switches, routers, etc.) connected by cables or wireless media for the exchange of information and resources. [CSTA, 2016]

Operation An action, resulting from a single instruction, that changes the state of data. [Free Dictionary]

Output Any device or component that receives information from a computer [Code.org]

Packet The unit of data sent over a network. [Tech Terms]

Password A password is a string of characters used to verify the identity of a user during the authentication process. Password is an example of one authentication factor. [TechTarget]

Parameter A special kind of variable used in a procedure to refer to one of the pieces of data received as input by the procedure. [MDESE, 2016]

Piracy The illegal copying, distribution, or use of software. [TechTarget]

Procedure An independent code module that fulfills some concrete task and is referenced within a larger body of program code. The fundamental role of a procedure is to offer a single point of reference for some small goal or task that the developer or programmer can trigger by invoking the procedure itself. [Techopedia] In this framework, procedure is used as a general term that may refer to an actual procedure or a method, function, or module of any other name by which modules are known in other programming languages.

Process A series of actions or steps taken to achieve a particular outcome. [Oxford]

Program (noun): A set of instructions that the computer executes to achieve a particular objective. [MDESE, 2016]; (verb): To produce a program by programming.

Programming The craft of analyzing problems and designing, writing, testing, and maintaining programs to solve them. [MDESE, 2016]

Protocol The special set of rules used by endpoints in a telecommunication connection when they communicate. Protocols specify interactions between the communicating entities. [TechTarget]

APPENDIX A: GLOSSARY for COMPUTER SCIENCE STANDARDS—page 5 of 5

Prototype A prototype is an early sample, model, or release of a product built to test a concept or process or to act as a thing to be replicated or learned from. [Wikipedia]

Redundancy A system design in which a component is duplicated, so if it fails, there will be a backup. [TechTarget]

Reliability Consistently produces the same results, preferably meeting or exceeding its requirements. [FOLDOC]

Remix The process of creating something new from something old. Originally a process that involved music, remixing involves creating a new version of a program by recombining and modifying parts of existing programs, and often adding new pieces, to form new solutions. [Kafai & Burke, 2014]

Router A device or software that determines the path that data packets travel from source to destination. [TechTarget]

Scalability The capability of a network to handle a growing amount of work or its potential to be enlarged to accommodate that growth. [Wikipedia]

Simulate To imitate the operation of a real-world process or system.

Simulation Imitation of the operation of a real-world process or system. [MDESE, 2016]

Software Programs that run on a computing system, computer, or other computing device.

Storage (noun): A place, usually a device, into which data can be entered, in which the data can be held, and from which the data can be retrieved at a later time. [FOLDOC]
storage (verb): A process through which digital data is saved within a data storage device by means of computing technology. Storage is a mechanism that enables a computer to retain data, either temporarily or permanently. [Techopedia]

String A sequence of letters, numbers, and/or other symbols. A string might represent, for example, a name, address, or song title. Some functions commonly associated with strings are length, concatenation, and substring. [TechTarget]

Structure A general term used in the framework to discuss the concept of encapsulation without specifying a particular programming methodology.

Switch A high-speed device that receives incoming data packets and redirects them to their destination on a local area network (LAN). [Techopedia]

System A collection of elements or components that work together for a common purpose. [TechTarget] See also the definition for computing system.

Test Case A set of conditions or variables under which a tester will determine whether the system being tested satisfies requirements or works correctly. [STF]

Topology The physical and logical configuration of a network; the arrangement of a network, including its nodes and connecting links. A logical topology is the way devices appear connected to the user. A physical topology is the way they are actually interconnected with wires and cables. [PCMag]

Troubleshooting A systematic approach to problem solving that is often used to find and resolve a problem, error, or fault within software or a computing system. [Techopedia, TechTarget]

Variable A symbolic name that is used to keep track of a value that can change while a program is running. Variables are not just used for numbers; they can also hold text, including whole sentences (strings) or logical values (true or false). A variable has a data type and is associated with a data storage location; its value is normally changed during the course of program execution. [CAS, 2013; Techopedia] Note: This definition differs from that used in math.