Washington Office of Superintendent of
PUBLIC INSTRUCTION

GUIDANCE ON TEACHING COMPUTER SCIENCE IN WASHINGTON STATE K–12 PUBLIC SCHOOLS

Authorizing legislation: HB 1577, SHB 5088

2020

GUIDANCE ON TEACHING COMPUTER SCIENCE IN WASHINGTON STATE K–12 PUBLIC SCHOOLS

2020

Kathe Taylor, Ph.D.
Assistant Superintendent
Learning and Teaching

Shannon Thissen
Computer Science Program Supervisor

Prepared by:

Janet Gordon, Ed.D.
10janetgordon@gmail.com | 509-860-5273

Washington Office of Superintendent of
PUBLIC INSTRUCTION

TABLE OF CONTENTS

Executive Summary 1
Introduction to the Guidance Document 2
Background 2
Washington State Definition of Computer Science 3
Similarities and Overlaps between Computer Science and Educational Technology 4
Foundational Knowledge for Computer Science 5
Washington State Learning Standards for Computer Science 6
Varied Instructional Settings to Teach Computer Science 11
Computer Science Integration into Various Content Areas 12
Next Steps 12
Support to Implement Computer Science in Your School 12
School Reporting of Computer Science Courses 13
Assessment of Computer Science Courses 14
Computer Science state Course Code Guidance 15
Computer Science STATE Course Codes 17
CTE CIP Codes and STATE Course Codes 18
Course Descriptions 24
Terms and Definitions 24
Standards and Practices by Grade Band Guide 36
Introduction 36
K–2 Standards & Practices 39
1. Fostering an Inclusive Computing Culture 39
2. Collaborating around Computing 40
3. Recognizing & Defining Computational Problems 41
4. Developing & Using Abstractions 42
5. Creating Computational Artifacts 47
6. Testing and Refining Computational Artifacts 49
7. Communicating About Computing 50
Grades 3–5 CS Standards & Practices 55
1. Fostering an Inclusive Computing Culture 55
2. Collaborating around Computing 57

3. Recognizing & Defining Computational Problems 58
4. Developing & Using Abstractions 60
5. Creating Computational Artifacts 62
6. Testing and Refining Computational Artifacts 64
7. Communicating About Computing 67
Grades 6–8 Standards & Practices 72
1. Fostering an Inclusive Computing Culture 72
2. Collaborating Around Computing 73
3. Recognizing & Defining Computational Problems 76
4. Developing and Using Abstractions 77
5. Creating Computational Artifacts 82
6. Testing and Refining Computational Artifacts 85
7. Communicating About Computing 88
Grades 9–10 Standards & Practices 91
1. Fostering an Inclusive Computing Culture 91
2. Collaborating Around Computing 93
3. Recognizing & Defining Computational Problems 95
4. Developing and Using Abstractions 97
5. Creating Computational Artifacts 103
6. Testing and Refining Computational Artifacts 108
7. Communicating About Computing 111
Appendices 116
Appendix A. Computer Science State Advisory Committee 116
Appendix B. Frequently Asked Questions—Superintendent. 117
Appendix C. Frequently Asked Questions—High School Principal. 119
Appendix D. Frequently Asked Questions—K–8 Principal. 121
Appendix E. Methodology used in the development of this document. 123
Legal Notice 124

EXECUTIVE SUMMARY

"The rise of Google, the rise of Facebook, the rise of Apple I think are proof that there is a place for computer science as something that solves problems that people face every day."[1] One of the most exciting and important aspects of this state initiative in computer science (CS) is that *all* students will have the opportunity to gain tangible benefits from engaging in computer science and solution-based practices. Recent legislation in Washington State requires that all high schools offer CS courses, which should begin with foundational experiences in kindergarten. Elementary and middle schools are the ideal places to ensure that every student has access to high quality foundational CS instruction.

To assist in the implementation of this requirement, OSPI assembled subject matter experts from higher education, Career and Technical Education, K–12, business, and other knowledgeable state representatives to create a definition of computer science specifically for the K–12 environment. The definition was then piloted in several schools. The refined definition presented in this document, is grounded in state and national standards and provides clarity around the question, "What does computer science look like in my classroom?"

Schools are encouraged to leverage their existing strengths to provide integrated and stand-alone CS courses in varied instructional settings to attract students from widely diverse backgrounds. State and national organizations offer approaches that are attractive to many students' interests, such as using CS to help their community or promote social justice around a particular need. Additionally, adaptive technologies can increase access to CS for students with disabilities and those served by a Section 504 plan. These solutions help students to achieve greater independence and productivity as well as expanded opportunities for positive social inclusion.

Under the legislation, high schools are required to report specific data, including the state course code, CIP code, and demographics of students enrolled in the courses. This comprehensive guide offers three documents to assist schools in implementing CS courses and reporting accurate data: (1) Computer Science Course Code Guidance; (2) Computer Science Course Descriptions; and (3) Computer Science Standards and Practices by Grade Band. Please continue to visit the OSPI website to keep up to date on information about the state initiative in computer science.

[1] Eric Schmidt, former Chief Executive Officer of Google

INTRODUCTION TO THE GUIDANCE DOCUMENT

The Office of Superintendent of Public Instruction (OSPI) offers this guidance to support schools to equitably teach computer science (CS) in Washington's K–12 public schools. This will assist districts to deeply understand and gain clarification on the new definition of computer science adopted by OSPI. OSPI is dedicated to providing a realistic and relevant definition of CS to guide school districts in determining the extent to which CS is being taught and guide districts' next steps. This definition will: (1) allow school districts the flexibility needed to accommodate their unique circumstances, (2) promote culturally responsive computing, (3) encourage the use of CS as a conduit to teach other content areas such as math and science, and (4) provide room for CS to evolve and include new technologies that have yet to be discovered.

BACKGROUND

In 2015, the Washington State Legislature directed the Superintendent of Public Instruction to adopt nationally recognized K–12 computer science standards (SHB 1813). The Computer Science K–12 Learning Standards Advisory Committee guided the selection and implementation of the K12CS.org framework and standards from the Computer Science Teachers Association (CSTA). OSPI's process of adoption included a comprehensive bias and sensitivity review by the Equity and Civil Rights Office before the statewide implementation of the new framework and standards.

The framework has provided educators with a scope and sequential progression for students to learn the academic content in the standards. However, districts' decision to teach computer science (CS) has been voluntary until the 2019 legislative session that required all high schools in Washington state to provide equitable access to CS and teach at least one CS course. Districts must also follow a structured method of reporting on students enrolled in the course, and guidance documents are attached.[2]

Working closely with a Computer Science Advisory Committee (Appendix A) assembled in 2019–20, Washington state developed a definition of CS to offer to districts that provide clarity around what qualifies as CS and what does not qualify as CS. Acknowledging that the spirit of the legislation is for *all* students to successfully engage with technology in both their personal and career life, the committee offered their expertise for districts to develop a healthy computer science "ecosystem" that offers access points to traditionally underserved and diverse students. This document offers the CS definition, guidance, and other informational resources for districts to comply with legislation and further support CS initiatives in every district across the state. Additionally, throughout this document, call-out boxes labeled "K–8 Focus" provide information to assist with elementary and middle school planning of CS instruction. There is also a Frequently Asked Questions section in the appendix for K–8 Principals. Lastly, the language used in this document is intentionally written to be accessible to a broad audience from novice to expert to meet the needs of districts across the state with a wide range of experience in CS instruction.

[2] HB 1577 SHB 5088

WASHINGTON STATE DEFINITION OF COMPUTER SCIENCE

The state definition of computer science includes, but is not limited to, the following ideas:

- The design of both computer equipment and digital systems, and the interface between the hardware and software required for these systems.
- How algorithms, data structures, and modules are used to implement computer software and hardware.
- Problem-solving skills for designing computer software and hardware such as pattern recognition, decomposition, debugging, and software troubleshooting.
- How hardware and software are used to implement computers, networks, and other digital systems.
- The use of computer programs to collect, analyze, store, transform, model, and visualize data.
- How networking devices enable communication and organization and increase the need for cybersecurity.
- Using computers to collect, analyze, transform and store data to create visualizations, models, and inferences.
- How the privacy and security of data can be protected with computers.
- How computers affect people and society.

Similarities and Overlaps between Computer Science and Educational Technology

There are similarities and overlaps between the definition of computer science and educational technology. Educational technology consists of educational technology literacy and its next level of skill development, technological fluency. In contrast, computer science is focused on the creation of computing hardware and software and its impact on society. Note that computer science does not include computer literacy education, which focuses on the use of existing technologies (e.g., word processing).

K–8 FOCUS—The definition above refers to computer science instruction at all grade levels. For more specific application of this definition to the K–8 setting, see Washington State Computer Science Standards and Practices, which details grade band appropriate learning and practices that can be applied in the classroom.

Educational technology literacy is the ability to:

- Responsibly, creatively, and effectively use appropriate technology to communicate.
- Access, collect, manage, integrate and evaluate information.
- Solve problems and create solutions.
- Build and share knowledge.
- Improve and enhance learning in all subject areas and experience.

Educational technology fluency is demonstrated when students:

- Apply technology to real-world experiences.
- Adapt to changing technologies.
- Modify current and create new technologies.
- Personalize technology to meet personal needs, interests, and learning styles[3].

[3] Office of Superintendent of Public Instruction (2018). K–12 Educational Technology Learning Standards, 98 pp.

FOUNDATIONAL KNOWLEDGE FOR COMPUTER SCIENCE

K–8 FOCUS—Early and consistent engagement in CS foundational practices across K–8 can boost confidence and enjoyment and prepare students to succeed in high school and beyond.

Computer science instruction needs to start in the early grades. Similar to all content areas, students need to acquire foundational computer science knowledge in elementary school to prepare them for success in the middle- and high-school computer science courses. This foundational knowledge leads to mastery of skills and abilities by delivering content that is sequential and follows established learning progressions (CSTA, 2017).

Foundational computer knowledge includes using and maintaining computer technology in safe and secure ways such as password protection, internet security, and online responsibility. All students also need to learn the appropriate and responsible use of technology among users called digital citizenship (ISTE, 2016). The three principles of digital citizenship are 1) respect, 2) educate, and 3) protect. These principles are essential for students to practice in school and beyond to remain safe in the digital world.

In elementary school, students begin to develop computational thinking skills by being guided to think through the processes to formulate problems and design solutions. Computational thinking skills help design a solution that can be executed by computers. Computational thinking skills are essential for proficiency in the CS standards as students learn increasingly complex content in middle and high school.

Foundational knowledge of how to collect, analyze, transform, present, store, and distribute data are all key bridges to the Common Core and Next Generation Science Standards (NGSS) throughout K–12. Creating simple computational artifacts such as an image, graphic, or audio recording prepares students to develop later more complex artifacts such as a video, web page, or program code (applications, games, etc.). Similarly, using a computer program to manipulate data, such as spreadsheets, also sets the stage for learning scripting languages or other programming languages.

An awareness of human-computer interaction (HCI) is also considered foundational knowledge. This awareness leads to an understanding of how culture and diversity affect the design of user interfaces and user experiences. Critical thinking about HCI enables students to design solutions and code interfaces while considering factors such as cultural relevance and accessibility.

Boosting foundational knowledge and providing encouragement beginning in the early grades can increase young students' confidence and enjoyment in computing. Gaining computer science knowledge early on can help dispel stereotypes and barriers related to diverse participation and engagement in computing and technology.

Washington State Learning Standards for Computer Science

The Washington State definition of computer science is aligned to the Washington State Learning Standards for CS. The standards are an essential resource and provide examples of vertically aligned CS practices that build on foundational knowledge in kindergarten and become increasingly complex as students increase their depth of understanding. An OSPI companion document called *Washington State Computer Science Standards and Practices by Grade Level Band* helps provide clarity by translating the standards into a familiar everyday language free from technical terms. The document also provides the core concept and core practice for each standard as well as samples of student performance.

Five overarching core concepts represent the major content areas in the field of CS. The core concepts are the knowledge areas associated with computer science are:

- Computing Systems
- Networks and the Internet
- Data and Analytics
- Algorithms and Programming
- Impacts of Computing

Seven core practices describe the behaviors and ways of thinking that students use when working within the core concepts. They are:

- Fostering an inclusive computing culture
- Collaborating around computing
- Recognizing and defining computational problems
- Developing and using abstractions
- Creating computational artifacts
- Testing and refining computational artifacts
- Communicating about computing

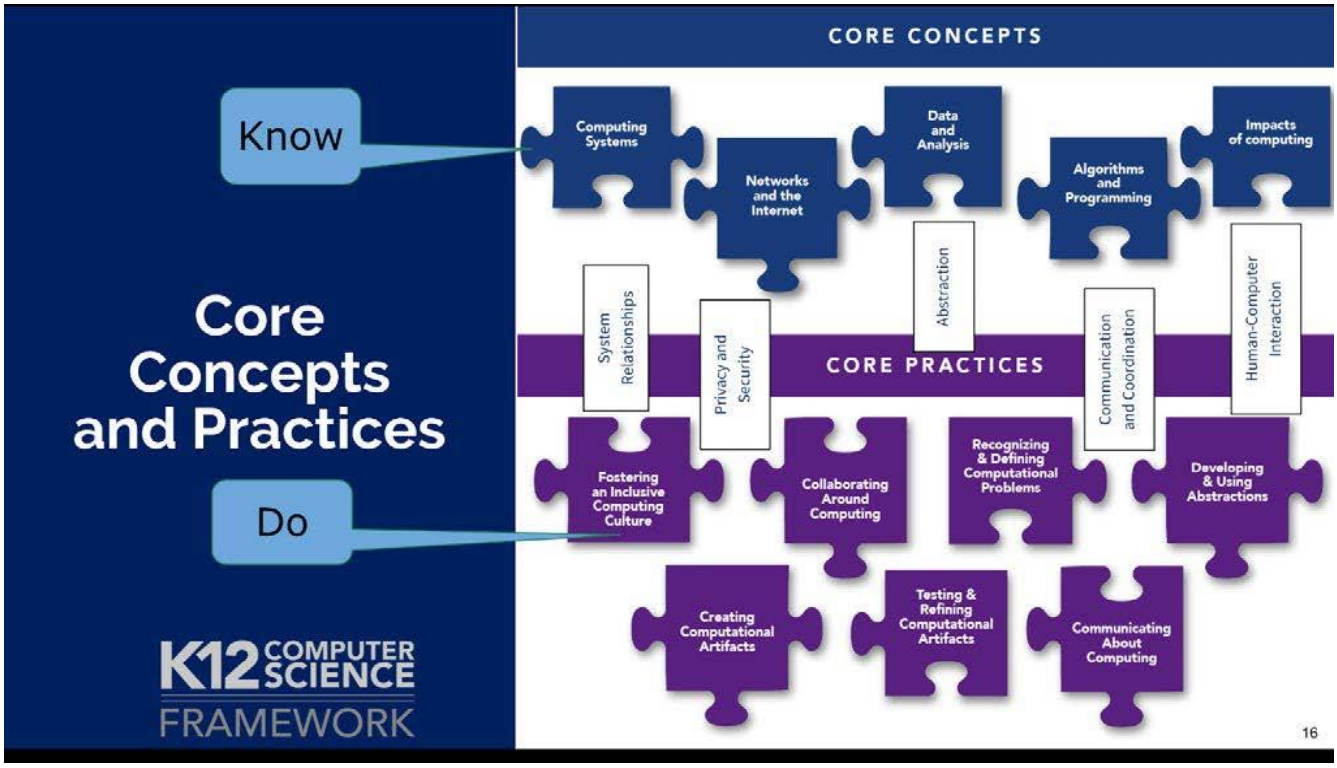


Figure 1: Five core concepts and seven core practices describe what students should know and do in CS.

Each CS learning standard consists of a core concept (what students need to do) and a core practice (how students need to do it). For middle and high school CS courses, examining the CS learning standards and aligning them to classroom instruction will guide districts through a process to choose an accurate Classification of Instructional Programs (CIP) code. Choosing the correct CIP code and state course code is critical to support accurate reporting and tracking of student enrollment.

K–8 FOCUS—Currently, K–6 grade levels may have a general state course code assigned to a grade level without delineating specific CS courses. However, elementary schools should try to identify any CS instruction and practice that can integrate into core classes. Middle Schools (grades 7–8) may also wish to more specifically define CS instruction integrated into core classes and select state course codes that better align with the course descriptions at the end of this document.

K–8 FOCUS—Currently, K–6 grade levels may have a general state course code assigned to a grade level without delineating specific CS courses. However, elementary schools should try to identify any CS instruction and practice that can integrated into core classes.

This intersection provides a definition of CS that can be used and applied within K–12 education[4] to teach and assess content (Figure).

Many school districts are already teaching CS to a certain extent. Schools may already embed some aspects of foundational computer science content across multiple areas and courses. Examples of these areas and courses may include CTE technology courses, educational technology, digital citizenship, internet safety, and digital literacy. Computer science instruction does not need to occur in a stand-alone class with separate content, especially at the elementary level. Instead, computer science instruction may be a part of inter-related instruction taught within and across the examples of areas and courses.

Figure 2: Key practices expressed within concepts lead to students doing computer science.

[4] K–12 Computer Science Framework. (2016). Retrieved from https://www.k12.cs.org

Equity in Computer Science Education

OSPI aims to equitably expand computer science education and broaden participation in computing across Washington State. To achieve this goal, we must first capture data that can be examined with a demographic breakdown including ethnicity, socioeconomic background, gender, etc. and evaluate (1) student access, (2) student participation, and (3) student outcomes and success with respect to CS education. CS must be offered equitably by deliberately engaging traditionally underrepresented students, including Latinx, African American, American Indian/Alaska Native, Pacific Islander, English Learners, girls, and students with disabilities. The ability to access a computer science course at a school is equity in its purest form. Some schools do not offer such a course yet; however, recent legislation requires that all high schools must offer a computer science course[5].

Washington has experienced growth in the diversity of our student populations and is a source of strength in our classrooms. CS can be relevant and attractive to diverse students through culturally responsive course content that considers *all* students' interests, experiences, and needs. Best practices for culturally responsive teaching include:

- Modify curriculum to be relevant to the lived experiences of the student.
- Use student strengths as a starting point and build on prior knowledge.
- Invest in and take personal responsibility for students' success.
- Create and nurture cooperative and collaborative environments.
- Encourage relationships among schools and communities.
- Promote critical literacy and thinking.
- Make explicit the power dynamics of mainstream society.
- Share power in the classroom.
- Engage students in social justice work [6].

[5] SHB 1577 SB 5088

[6] Culturally Responsive Teaching, Region 10 Equity Assistance Center, 2009.

Districts are encouraged to offer meaningful computing. Students who are engaged in real-life issues (i.e., environment, animal welfare, health, and nutrition) in the context of their own communities are more motivated and persistent in CS education. National organizations such as CSforAll, Girls that Code, National Center for Women in Technology (NCWIT), Black Boys Code, Black Girls Code, etc. all offer resources to help a district adapt and add course content that motivates and engages traditionally underserved students.

Bias and cultural sensitivity is a critical consideration when selecting, adapting, or modifying the CS curriculum and learning resources. OSPI's Equity and Civil Rights Office has established guidelines to assist in the review of bias in learning resources (Table 1).

Table 1. OSPI Equity and Civil Rights Office guidelines for a review of bias in learning resources[7].

A Bias Review Should Consider the Following Elements		
Gender	Race	Ethnicity
Sexual Orientation	Religion	Socio-Economic Status
Gender Expression & Identity	Physical Disability	Age
Family Structure	Native Language	Occupation
Body Shape/Size	Culture	Geographic Setting

Equitable access to CS can improve for students with learning and attention disabilities. Educators can use a variety of tools and strategies to ensure that learners of all abilities are successful. These strategies can, when incorporated into the classroom, including pedagogical strategies and assistive and adaptive technologies that can enhance the educational experience of students with disabilities and those served by a Section 504 plan. These solutions help students to achieve greater independence and productivity as well as expanded opportunities for positive social inclusion.

[7] Retrieved from https://www.k12.wa.us/policy-funding/equity-and-civil-rights/reviewing-instructional-materials-bias

VARIED INSTRUCTIONAL SETTINGS TO TEACH COMPUTER SCIENCE

There is a wide range of ways that CS can be offered in the K–12 environment to increase accessibility to traditionally underserved students. In middle and high school, CS can be offered as a unique course or may be taught in coordination with other areas of the curriculum, such as math, science, technology, and business. In the early grades, foundational skills may be offered through a variety of expanded learning opportunities (ELOs), including during the school day, in after-school programs, and inter-session programs. ELOs provide educational supports as well as enrichment across a wide range of social contexts that maybe school- or community-based.

As always, achieving equitable access to CS is the goal, and districts can help by mitigating barriers and providing the support needed for all students to participate. This requires districts to engage in deliberate strategies to increase the participation of underserved students, including girls, diverse ethnicities, low-income students, and students with disabilities, as well as involve family members.

K–8 FOCUS—The goal to increase equity in CS instruction in high schools is supported by comprehensive CS instruction. Elementary and Middle Schools are the ideal places to ensure that every student has access to high quality foundational CS instruction.

Schools are encouraged to leverage existing strengths and build on CS instruction that is already in place and set equity-based strategic goals to expand access to CS instruction to meet the state requirements. Schools may be positioned to expand current CS opportunities through a redistribution of resources in a manner that maximizes student access and meets the needs of all students. In many cases, this expansion can be accomplished by embedding CS as a conduit to teach other subjects within various academic disciplines. For example, a science class may include the use of instrumentation to collect data for lab experiments where the instrumentation may require programming. Large amounts of raw data may be generated that require manipulation and analysis to convert into usable information.

Additional ideas are located in the Next Generation Science Standards, where practices such as computational thinking are described within the standards[8]. CS courses may be offered as stand-alone elective credit or may be available for dual credit with instruction in other academic areas. The instructional settings to teach computer science can be varied; however, the content must be linked to standards and be mastery-based, with an assessment of learning.

[8] https://www.nextgenscience.org/search/node/computational_thinking

COMPUTER SCIENCE INTEGRATION INTO VARIOUS CONTENT AREAS

CS integration is possible throughout K–12. For example, when students develop a simple programming game to practice math skills, they are learning both math and CS. When middle school biology students develop an algorithm to summarize data in a research project, they are learning about biology as well as the computational tools to analyze the data. These types of multi-discipline approaches to CS instruction can be essential components in the school's CS curriculum.

A primary goal for schools is to offer multiple ways to attract students from a wide variety of cultural backgrounds so they can see that pursuing technology education and careers is within reach. For example, broadening participation to girls, and diverse students requires expanded learning opportunities that are relevant to them, such as using CS to help their community or promote social justice around a particular need (e.g., CSforgood.org). Also, CS can be scheduled in co-curricular settings such as Robotics Clubs and After School Projects. Schools can encourage students' participation in competitions and exhibitions, a requirement in CTE courses.

NEXT STEPS

Support to Implement Computer Science in Your School

OSPI, in partnership with the Education Service Districts (ESDs), offers assistance to schools for strategic planning to build and implement a cohesive, robust CS program. There are many approaches to strategic planning, one of which is the Strategic CSforALL Resource and Implementation Planning Tool (SCRIPT). SCRIPT is a framework and systematic process that engages school staff, community, and stakeholders in collaborative activities to identify the components needed to implement a CS program successfully. Through the SCRIPT process, the school is guided through five focused areas: (1) materials and curriculum selection and content refinement, (2) leadership, (3) teacher capacity and development, (4) partners, and (5) community.

In a reiterative cycle, participants in SCRIPT training identify where they are as a school district (novice, emerging, developing, highly developed) in the focused areas. In the first focused area, materials/curriculum selection and content refinement, participants discuss elements of high-quality computer science curriculum including inclusivity, multidisciplinary, scaffolded, and sequential with teacher supports such as formative/summative assessments.

SCRIPT trainers will facilitate discussions around the critical leadership needed to implement a CS program successfully. Success relies on leaders who can cast the vision, set goals, and plan how to carry the program out. Successful leaders can motivate others across the district to share the responsibility to reach their goals. School personnel such as librarians, special educators, and guidance counselors all bring different perspectives and can help reach out to parents and promote community engagement in the vision.

Teacher capacity building and professional development is a significant determinant of successfully reaching district goals to deliver effective CS instruction. The SCRIPT process guides participants to incorporate teacher professional development into their school plan as well as local, state, and national partners that can help to bring high-quality CS courses and provide professional development to teachers. Lastly, SCRIPT training raises awareness of how the community plays an integral part in the development and sustainability of a CS program. This awareness can lead to improved school-to-family communication about partner opportunities, extracurricular activities, and in-school CS pathways. The community can also help inform the CS plan and pathways based not only for college but also for career readiness based on the local workforce needs.

SCHOOL REPORTING OF COMPUTER SCIENCE COURSES

As all schools begin to offer K–12 CS classes, OSPI is committed to building an understanding of the support that districts will need to offer CS classes successfully. This deeper understanding is dependent upon the accuracy of the information that school districts report. Schools must report CS courses offered, students enrolled, and staff instructors. The student-record system must include state course codes for CS and enrollment demographics. Teacher certification reports must include demographics regarding instructors for CS courses.

To increase reporting data accuracy, OSPI has developed documentation that offers CS course code guidance. This documentation can be found below in the next section. This resource aligns the core concepts and practices to CS course names, descriptions, and Classification of Instructional Programs (CIP) codes. This will assist districts in aligning course content to a course name, CIP code, and the number of credits. Choosing the correct CIP code is critical to support accurate reporting and tracking of enrollment demographics. The course data will be analyzed across the state to more deeply understand the support schools need to implement CS at the K–12 level.

ASSESSMENT OF COMPUTER SCIENCE COURSES

OSPI requires that a substantial percent of the Washington State Computer Science Learning standards are taught and assessed in each credit-bearing CS course. The CS standards may be taught through the integration of other content such as math, science, or ELA. OSPI is currently working on developing the rules for oversight of the assessments that award CS credit. In addition, high schools must offer competency testing that demonstrates mastery of CS standards. CS competency exams must be able to award CS credit for skills acquired through vocational courses and work experience.

K–8 FOCUS—Assessment of CS at the K–8 level is conducted through locally developed classroom assessments. These assessments are used by instructional staff to determine student mastery of learning targets.

COMPUTER SCIENCE STATE COURSE CODE GUIDANCE

During the 2019–20 Legislative session, SHB 1577 concerning K–12 computer science education data was passed into law. Beginning June 30, 2020, and by June 30 annually after that, school districts must submit to the Office of the Superintendent of Public Instruction (OSPI), and the OSPI must post conspicuously on its website, a report for the preceding academic year that must include the following data:

- Total number of computer science courses offered in each school and whether these courses are advanced placement classes.
- Number and percentage of students who enrolled in a computer science program.
- Disaggregated by gender, race and ethnicity, special education status, English learner status, eligibility for the free and reduced-price lunch program, and grade level.
- Number of computer science instructors at each school, disaggregated by certification, if applicable, gender, and highest academic degree.

Data collection will be done through the Comprehensive Education Data and Research System (CEDARS), a longitudinal data system managed by the Office of the Superintendent of Public Instruction (OSPI) to collect, store, and report data related to students, courses, and teachers. The data collected is either mandated by state or federal law or approved by the Data Governance Group at the OSPI.

CEDARS contains a course catalog of all courses in each grade offered at each public school. Student-related information in CEDARS includes each student's gender, grade level, demographics, eligibility for specific education programs, and a record of all courses attempted by the student. For students in grades 9 through 12, final grades and credit information for each course attempted and earned by the student are also stored in CEDARS. There is also information in CEDARS about the staff member teaching each course or assigned to a homeroom, including each staff member's gender, academic degrees, and certification.

State Course Codes are reported within the Comprehensive Education Data and Research System (CEDARS) and were developed using the National Center for Educational Statistics (NCES) course codes. Reporting State Course Codes are required for all courses reported to CEDARS. Local education agencies (LEAs) determine the state course code most appropriate for each class offered. Course information is amended with data populated from CEDARS.

Data to fulfill the legislation will be directly retrieved from CEDARS. For the data to be accurate, school districts must code their Computer Science courses with the correct state course code. The following list is the courses that will count as Computer Science courses in fulfilling the legislative intent of SHB5088 requiring all comprehensive high schools to offer a Computer Science course by the 2022–23 school year.

Table 1 of this document lists the state course codes that will meet the legislative requirement.

Table 2 contains the CTE CIP codes and recommended state course codes. If you are offering any courses using the following CIP codes, please review the State Course Code and Course Name in the table. Based on your framework submitted under that CIP code, please use the appropriate State Course Code according to the Course Name in the table. If done correctly, this will allow OSPI to report the requested data in above mentioned SHB 1577. So, if you are using any of the CIP codes listed and have an appropriate State Course code from this table, your district will meet the criteria of the legislation.

Table 3 contains the Course Descriptions to help determine where the course you are teaching the best fits.

COMPUTER SCIENCE STATE COURSE CODES

*** New State Course Codes Starting 2021–22**

Table 2: Computer Science State Course Codes

State Course Code	Course Name
10011	Computer Science Principles
10012	Exploring Computer Science
10013	**PLTW Computer Science Essentials***
10014	**PLTW Computer Science A***
10015	**PLTW Computer Science Principles***
10016	**PLTW Cybersecurity***
10019	AP Computer Science Principles
10020	**Cybersecurity***
10052	Database Management and Data Warehousing
10053	Database Applications
10054	Data Systems/Processing
10097	Management Information Systems—Independent Study
10098	Management Information Systems—Workplace Experience
10099	**Information Technology-Other***
10101	Network Technology
10102	Networking Systems
10108	Network Security
10109	Essentials of Network Operating Systems
10148	Networking Systems—Workplace Experience
10149	Networking System – other
10152	Computer Programming
10153	Visual Basic (VB) Programming

State Course Code	Course Name
10154	C++ Programming
10155	Java Programming
10156	Computer Programming—Other Language
10157	AP Computer Science A
10159	IB Computer Science
10160	Particular Topics in Computer Programming
10197	Computer Programming Independent Study
10198	Computer Programming— Workplace Experience
10199	Computer Programming — Other
10201	Web Page Design
10203	Interactive Media
10205	Computer Gaming and Design
10206	Mobile Applications
10251	Computer Technology
10253	Information Support and Services
10254	IT Essentials: PC Hardware and Software
10297	Information Support Services Independent Study
10298	Information Support and Services—Workplace Experience
10301	**Computer Forensics***

CTE CIP CODES AND STATE COURSE CODES

*** New State Course Codes Starting 2021–22**

♦ Not Computer Science
Theses course are information technology courses

Table 3: CTE CIP Codes and State Course Codes

CIP Code	Teacher Cert V-Code	State Course Code	Course Name	SUBJECT	Course Description
110201 Preparatory	V070000 V078000 V141000 V210100 V470110 V521206	10011	Computer Science Principles	Computer Programming Brainbench	A course that focuses on the general writing and implementation of generic and customized programs to drive operating systems and that generally prepares individuals to apply the methods and procedures of software design and programming to software installation and maintenance. Includes instruction in software design, low- and high-level languages and program writing; program customization and linking; prototype testing; troubleshooting; and related aspects of operating systems and networks.
		10014	***PLTW Computer Science A**		
		10015	***PLTW Computer Science Principles**		
		10019	AP Computer Science Principles		
		10152	Computer Programming		
		10153	Visual Basic (VB) Programming		
		10154	C++ Programming		
		10155	Java Programming		
		10156	Computer Programming—Other Language		
		10157	AP Computer Science A		
		10159	IB Computer Science		
		10197	Computer Programming—Independent Study		
		10199	Computer Programming— Other		

CIP Code	Teacher Cert V-Code	State Course Code	Course Name	SUBJECT	Course Description
110204 Preparatory	V070000 V078000 V141000 V210100 V470110 V521206	10205	Computer Gaming and Design	Computer Game Programming	A program that prepares individuals to apply the knowledge and skills of design and computer programming to the development of computer games. Includes training in character and story development, computer programming, computer graphics, game design, game physics, human-computer interaction, human-centered design, and usability.
		10253	Information Support and Services		
110701 Exploratory	V07000 V07800 V141000 V210100 V521206 V470110	10012	Exploring Computer Science	Introduction to Computer Science	A program that focuses on computer theory, computing problems and solutions, and the design of computer systems and user interfaces from a scientific perspective. Includes instruction in the principles of computational science, computer development and programming, and applications to a variety of end-use situations.
		10013	***PLTW Computer Science Essentials**		
		10152	Computer Programming		
		10160	Particular Topics in Computer Programming		
110801 Preparatory	V070000 V078000 V100100 V470110 V521206	10201	Web Page Design	Webpage/ Digital/ Multimedia and Information Design CIW Foundations	A course that prepares individuals to apply HTML, XML, JavaScript, graphic applications, and other authoring tools to the design, editing, and publishing (launching) of documents, images, graphics, sound, and multimedia products on the World Wide Web. Includes instruction in internet theory; web page standards and policies; elements of web page design; user interfaces; vector tools; special effects; interactive and multimedia components; search engines; navigation; morphing; e-commerce tools; and emerging web technologies.
		10203	Interactive Media		
		11151	**♦Digital Media Technology**		
		11153	**♦Digital Media Design and Production**		

CIP Code	Teacher Cert V-Code	State Course Code	Course Name	SUBJECT	Course Description
110802 Preparatory	V070000 V078000 V470110 V521206	10052	Database Management and Data Warehousing	Data Modeling and Database Administration MCDBA	A course that prepares individuals to design and manage the construction of databases and related software courses and applications, including the linking of individual data sets to create complex searchable databases (warehousing) and the use of analytical search tools (mining). Includes instruction in database theory, logic, and semantics; operational and warehouse modeling; dimensionality; attributes and hierarchies; data definition; technical architecture; access and security design; integration; formatting and extraction; data delivery; index design; implementation problems; planning and budgeting; and client and networking issues.
		10053	Database Applications		
		10054	Data Systems/Processing		
110803 Preparatory	V070000 V078000 V100100 V470110 V480101 V521206	**10202**	**♦Computer Graphics**	Video Game Design/Digital Computer Animation for Game Design Skill Connect Assessment	A course that focuses on the software, hardware, and mathematical tools used to represent, display, and manipulate topologically, two and three-dimensional objects on a video screen and prepares individuals to function as computer graphics/video game development specialists. Includes instruction in graphics software and systems; computer programming; digital multimedia; graphic design, video game design and development; graphics devices, processors, and standards; attributes and transformations; projections; surface identification and rendering; color theory; algebra; geometry; trigonometry and introduction to various mathematical concepts related to interactive computer and computer graphic-based applications.
		10203	Interactive Media		
		10205	Computer Gaming and Design		
		10206	Mobile Applications		

CIP Code	Teacher Cert V-Code	State Course Code	Course Name	SUBJECT	Course Description
110901 Preparatory	V070000 V078000 V470110 V470101	10101	Network Technology	Computer Systems Networking and Telecommunications Network +	A course that focuses on the design, implementation, and management of linked systems of computers, peripherals, and associated software to maximize efficiency and productivity, and that prepares individuals to function as network specialists and managers at various levels. Includes instruction in operating systems and applications; systems design and analysis; networking theory and solutions; types of networks; network management and control; network and flow optimization; security; configuring; and troubleshooting. The second 180 hours of this course should lead to industry certification such as Cisco Certified Network Associate (CCNA) certification.
		10102	Networking Systems		
		10108	Network Security		
		10109	Essentials of Network Operating Systems		
111003 Preparatory	V070000 V078000 V141000 V210100 V470110 V521206	**10016**	***PLTW Cybersecurity**	Computer and Information Systems Security/Auditing /Information Assurance	A program that prepares individuals to assess the security needs of computer and network systems, recommend safeguard solutions, and manage the implementation, auditing, and maintenance of security devices, systems, and procedures. Includes instruction in computer architecture, programming, and systems analysis; networking; telecommunications; cryptography; security system auditing and design; applicable law and regulations; risk assessment and policy analysis; contingency planning; user access issues; investigation techniques; and troubleshooting.
		10020	***Cybersecurity**		

CIP Code	Teacher Cert V-Code	State Course Code	Course Name	SUBJECT	Course Description
111004 Preparatory	V100100 V470110 V070000 V078000	10201	Web Page Design	Web/ Multimedia Management and Webmaster CIW Foundations	A program that prepares individuals to develop and maintain web servers and the hosted web pages at one or a group of web sites, and to function as designated webmasters. Includes instruction in computer systems and editing; information resources management; web policy and procedures; Internet applications of information systems security; user interfacing and usability research; and relevant management and communications skills.
		10202	**♦Computer Graphics**		
		10203	Interactive Media		
111006 Preparatory	V070000 V078000 V470110 V521206	10251	Computer Technology	Computer Support Specialist A+	A course that prepares individuals to analyze problems and research solutions; identify, test, and implement solutions; manage working relationships with customers; install, configure, and test new operating and application software and software upgrades; operate a computer system and run system applications; and monitor and analyze system performance. Includes instruction in troubleshooting; facilitation and customer service; hardware and software installation, configuration, and upgrades; and system operations, monitoring, and maintenance.
		10253	Information Support and Services		
		10254	IT Essentials: PC Hardware and Software		
		10301	Computer Forensics		
118888 Exploratory	V600096 V600097	10098	Management Information Systems—Workplace Experience	Computer and Information Sciences and Support Services Cooperative Worksite Experience	A learning experience in which the student has completed a Career and Technical Education sequence in their T&I Pathway education prior to the co-op experience or concurrently enrolls in a Career and Technical Education class at school and works in a related occupation. WAC 392-410-315 outlines regulations for granting credit for cooperative work-based learning experiences.
		10149	Networking Systems—Workplace Experience		
		10199	Computer Programming—Workplace Experience		
		10248	**♦Media Technology—Workplace Experience**		
		10998	**♦Information Technology—Workplace Experience**		

CIP Code	Teacher Cert V-Code	State Course Code	Course Name	SUBJECT	Course Description
151202 Preparatory	V470110	10251	Computer Technology	Computer Technology/ Computer Systems Technology	A program that prepares individuals to apply basic engineering principles and technical skills in support of professionals who use computer systems. Includes instruction in basic computer design and architecture, programming, problems of specific computer applications, component, and system maintenance, and inspection procedures, hardware and software problem diagnosis and repair, and report preparation.
		12053	Information Support and Services		

TERMS AND DEFINITIONS

Washington State has adopted the following definitions for clarity and direction of courses and curriculum.

Computational Thinking: A way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science, a fundamental skill for everyone, not just computer scientists.

Computing Education: The study of computer science or related activities, includes the act of scripting, coding, web development, or computer programming. It does *not* include non-coding uses of computer technology to create artifacts (i.e., multimedia development, desktop publishing).

Digital Citizenship: Digital citizens recognize and value the rights, responsibilities, and opportunities of living, learning, and working in an interconnected digital world, and they engage in safe, legal, and ethical behaviors.

Digital Literacy: An individual's ability to find, evaluate, and compose clear information through writing and other media on various digital platforms.

Educational Technology: Washington State's EdTech Standards define technology literacy and its next level of skill development, technological fluency, in this way:

> **Technology Literacy** is the ability to responsibly, creatively, and effectively use appropriate technology to:
>
> - Communicate.
> - Access, collect, manage, integrate, and evaluate information.
> - Solve problems and create solutions.
> - Build and share knowledge.
> - Improve and enhance learning in all subject areas and experiences.
>
> **Technology Fluency** is demonstrated when students:
>
> - Apply technology to real-world experiences.
> - Adapt to changing technologies.
> - Modify current and create new technologies.
> - Personalize technology to meet personal needs, interests, and learning styles.

Keyboarding: The goal of keyboarding is to enable proficient and accurate digital input. By grade 5, it is recommended that students should be able to key by touch. Key by touch is determined through proficiency, proper technique with associated keys and fingers, and not speed.

Media Literacy: The ability to access, analyze, evaluate, create, and act using a variety of forms of communication.

Course Descriptions

*** New State Course Codes Starting 2021–22**

Table 4: Course Descriptions

Course Code	Course Name	Description	Computer Science Standards
10011	Computer Science Principles	Computer Science Principles courses provide students the opportunity use programming, computational thinking, and data analytics to create digital artifacts and documents representing design and analysis in areas including the Internet, algorithms, and the impact that these have on science, business, and society. Computer Science Principles courses teach students to use computational tools and techniques including abstraction, modeling, and simulation to collaborate in solving problems that connect computation to their lives.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming 5. Impacts of Computing
10012	Exploring Computer Science	Exploring Computer Science courses present students with the conceptual underpinnings of computer science through an exploration of human computer interaction, web design, computer programming, data modeling, and robotics. While these courses include programming, the focus is on the computational practices associated with doing computer science, rather than just a narrow focus on coding, syntax, or tools. Exploring Computer Science courses teach students the computational practices of algorithm design, problem solving, and programming within a context that is relevant to their lives.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming 5. Impacts of Computing
10013	***PLTW Computer Science Essentials**	Following Project Lead the Way's suggested curriculum, PLTW Computer Science Essentials (formerly known as PLTW Introduction to Computer Science) courses introduce students to computational thinking concepts, fundamentals, and tools. Students will increase their understanding of programming languages through the use of visual and text-based programming. Projects will include the creation of apps and websites to address real-life topics and problems.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming 5. Impacts of Computing

Course Code	Course Name	Description	Computer Science Standards
10014	***PLTW Computer Science A**	Following Project Lead the Way's suggested curriculum to prepare students for the College Board's Advanced Placement Computer Science A exam, PLTW Computer Science A (formerly known as PLTW Computer Science Applications) courses focus on extending students' computational thinking skills through the use of various industry-standard programming and software tools. In these courses, students collaborate to design and produce solutions to real-life problems.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming 5. Impacts of Computing
10015	***PLTW Computer Science Principles**	Following Project Lead the Way's suggested curriculum to prepare students for the College Board's Advanced Placement Computer Science Principles exam, PLTW Computer Science Principles (formerly known as PLTW Computer Science and Software Engineering) courses are designed to help students develop computational thinking, and introduce students to possible career paths involving computing. These courses help students build programming expertise and familiarity with the Internet using multiple platforms and programming languages. Course content may include application development, visualization of data, cybersecurity, and simulation.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming 5. Impacts of Computing
10016	***PLTW Cybersecurity**	Following Project Lead the Way's suggested curriculum, PLTW Cybersecurity courses introduce students to the tools and concepts of cybersecurity. In these courses, students are encouraged to understand vulnerabilities in computational resources and to create solutions that allow people to share computing resources while retaining privacy. These courses also introduce students to issues related to ethical computing behavior.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming 5. Impacts of Computing

Course Code	Course Name	Description	Computer Science Standards
10019	AP Computer Science Principles	Following the College Board's suggested curriculum designed to parallel college-level computer science principles courses, AP Computer Science Principles courses introduce students to the fundamental ideas of computer science and how to apply computational thinking across multiple disciplines. These courses teach students to apply creative designs and innovative solutions when developing computational artifacts. These courses cover such topics as abstraction, communication of information using data, algorithms, programming, the Internet, and global impact.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming 5. Impacts of Computing
10020	***Cybersecurity**	Cybersecurity courses introduce students to the concepts of cybersecurity. These courses provide students with the knowledge and skills to assess cyber risks to computers, networks, and software programs. Students will learn how to create solutions to mitigate cybersecurity risks. These courses may also cover the legal environment and ethical computing behavior related to cybersecurity.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming 5. Impacts of Computing
10052	Database Management and Data Warehousing	Database Management and Data Warehousing courses provide students with the skills necessary to design databases to meet user needs. Courses typically address how to enter, retrieve, and manipulate data into useful information. More advanced topics may cover implementing interactive applications for common transactions and the utility of mining data.	CS includes subjects and standards in these core areas: 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet
10053	Database Applications	Database Application courses provide students with an understanding of database development, modeling, design, and normalization. These courses typically cover such topics as SELECT statements, data definition, manipulation, control languages, records, and tables. In these courses, students may use Oracle WebDB, SQL, PL/SQL, SPSS, and SAS and may prepare for certification.	CS includes subjects and standards in these core areas: 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet

Course Code	Course Name	Description	Computer Science Standards
10054	Data Systems/Processing	Data Systems/Processing courses introduce students to the uses and operation of computer hardware and software and to the programming languages used in business applications. Students typically use BASIC, COBOL, and RPL languages as they write flowcharts or computer programs and may also learn data-processing skills.	CS topics include subjects and standards in these core areas: 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet
10097	Management Information Systems—Independent Study	Management Information Systems—Independent Study courses, often conducted with instructors as mentors, enable students to explore topics related to management information systems. Independent Study courses may serve as an opportunity for students to expand their expertise in a particular specialization, to explore a topic in greater detail, or to develop more advanced skills.	CS topics include subjects and standards in these core areas: 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet
10098	Management Information Systems—Workplace Experience	Management Information Systems—Workplace Experience courses provide work experience in fields related to management information systems. Goals are typically set cooperatively by the student, teacher, and employer (although students are not necessarily paid). These courses may include classroom activities as well, involving further study of the field or discussion regarding experiences that students encounter in the workplace.	CS topics include subjects and standards in these core areas: 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet
10101	Network Technology	Network Technology courses address the technology involved in the transmission of data between and among computers through data lines, telephone lines, or other transmission media, such as hard wiring, wireless, cable networks, and so on. These courses may emphasize the capabilities of networks, network technology itself, or both. Students typically learn about network capabilities and network technology, including the software, hardware, and peripherals involved in setting up and maintaining a computer network.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming May include these areas: 5. Impacts of Computing

Course Code	Course Name	Description	Computer Science Standards
10102	Networking Systems	Networking Systems courses are designed to provide students with the opportunity to understand and work with hubs, switches, and routers. Students develop an understanding of LAN (local area network), WAN (wide area network), wireless connectivity, and Internet-based communications (including cloud-based computing), with a strong emphasis on network function, design, and installation practices. Students acquire skills in the design, installation, maintenance, and management of network systems that may help them obtain network certification.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming May include these areas: 5. Impacts of Computing
10108	Network Security	Network Security courses provide students with an understanding of network security principles and implementation. Course topics usually include authentication, the types of attacks and malicious code that may be used against computer networks, the threats and countermeasures for e-mail, Web applications, remote access, and file and print services. These courses may also cover a variety of security topologies as well as technologies and concepts used for providing secure communication channels, secure internetworking devices, intrusion detection systems, and firewalls.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet
10109	Essentials of Network Operating Systems	Essentials of Network Operating Systems courses provide students with an overview of multi-user, multi-tasking network operating systems. In these courses, students study the characteristics of operating systems, such as Linux, and various Windows network operating systems and explore a range of topics including installation procedures, security issues, back-up procedures, and remote access. Advanced topics may include network administration, including account management, training, evaluating new technology, developing system policies, troubleshooting, email and business communications and Web site management.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet

Course Code	Course Name	Description	Computer Science Standards
10148	Networking Systems—Workplace Experience	Networking Systems—Workplace Experience courses provide students with work experience in fields related to networking systems. Goals are typically set cooperatively by the student, teacher, and employer (although students are not necessarily paid). These courses may include classroom activities as well, involving further study of the field or discussion regarding experiences that students encounter in the workplace.	CS topics include subjects and standards in these core areas: 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming
10149	Networking System – other	Other Networking Systems courses.	
10152	Computer Programming	Computer Programming courses provide students with the knowledge and skills necessary to construct computer programs in one or more languages. Computer coding and program structure are often introduced with the BASIC language, but other computer languages, such as Visual Basic (VB), Java, Pascal, C++, and C#, may be used instead. Students learn to structure, create, document, and debug computer programs. Advanced courses may include instruction in object-oriented programming to help students develop applications for Windows, database, multimedia, games, mobile and Web environments. An emphasis is placed on design, style, clarity, and efficiency. In these courses, students apply the skills they learn to relevant authentic applications.	CS topics include subjects and standards in these core areas: 1. Computing Systems 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet 5. Impacts of Computing
10153	Visual Basic (VB) Programming	Visual Basic (VB) Programming courses provide an opportunity for students to gain expertise in computer programs using the Visual Basic (VB) language. As with more general computer programming courses, the emphasis is on how to structure and document computer programs and how to use problem-solving techniques. These courses cover such topics as the use of text boxes, scroll bars, menus, buttons, and Windows applications. More advanced topics may include mathematical and business functions and graphics.	CS topics include subjects and standards in these core areas: 1. Computing Systems 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet 5. Impacts of Computing

Course Code	Course Name	Description	Computer Science Standards
10154	C++ Programming	C++ Programming courses provide an opportunity for students to gain expertise in computer programs using the C++ language. As with more general computer programming courses, the emphasis is on how to write logically structured programs, include appropriate documentation, and use problem-solving techniques. More advanced topics may include multi-dimensional arrays, functions, sorting, loops, and records.	CS topics include subjects and standards in these core areas: 1. Computing Systems 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet 5. Impacts of Computing
10155	Java Programming	Java Programming courses provide students with the opportunity to gain expertise in computer programs using the Java language. As with more general computer programming courses, the emphasis is on how to structure and document computer programs, using problem-solving techniques. Topics covered in the course include syntax, I/O classes, string manipulation, and recursion.	CS topics include subjects and standards in these core areas: 1. Computing Systems 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet 5. Impacts of Computing
10156	Computer Programming—Other Language	Computer Programming—Other Language courses provide students with the opportunity to gain expertise in computer programs using languages other than those specified (such as Pascal, FORTRAN, Python, or emerging languages). As with other computer programming courses, the emphasis is on how to structure and document computer programs, using problem-solving techniques. As students advance, they learn how to utilize best the features and strengths of the language being used.	CS topics include subjects and standards in these core areas: 1. Computing Systems 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet 5. Impacts of Computing
10157	AP Computer Science A	Following the College Board's suggested curriculum designed to mirror college-level computer science courses, AP Computer Science A courses emphasize object-oriented programming methodology with a focus on problem solving and algorithm development. These courses cover such topics as object-oriented program design; program implementation; program analysis; standard data structures; standard algorithms; and the ethical and social implications of computing systems.	CS topics include subjects and standards in these core areas: 1. Computing Systems 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet 5. Impacts of Computing

Course Code	Course Name	Description	Computer Science Standards
10159	IB Computer Science	IB Computer Science courses prepare students to take the International Baccalaureate Computer Science exams. The courses emphasize system fundamentals, computer organization, and networks, as well as the fundamental concepts of computational thinking, the development of practical computational solutions, and programming. IB Computer Science courses also cover the applications and effects of the computer on modern society as well as the limitations of computer technology.	CS topics include subjects and standards in these core areas: 1. Computing Systems 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet 5. Impacts of Computing
10160	Particular Topics in Computer Programming	These courses examine particular topics in computer programming other than those already described elsewhere in this classification system.	
10197	Computer Programming—Independent Study	Computer Programming—Independent Study courses, often conducted with instructors as mentors, enable students to explore topics related to computer programming. Independent Study courses may serve as an opportunity for students to expand their expertise in a particular specialization, to explore a topic in greater detail, or to develop more advanced skills.	CS topics include subjects and standards in these core areas: 1. Computing Systems 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet 5. Impacts of Computing
10198	Computer Programming—Workplace Experience	Computer Programming—Workplace Experience courses provide students with work experience in fields related to computer programming. Goals are typically set cooperatively by the student, teacher, and employer (although students are not necessarily paid). These courses may include classroom activities as well, involving further study of the field or discussion regarding experiences that students encounter in the workplace.	CS topics include subjects and standards in these core areas: 1. Computing Systems 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet 5. Impacts of Computing

Course Code	Course Name	Description	Computer Science Standards
10199	Computer Programming — Other	Other Computer Programming courses.	CS topics include subjects and standards in these core areas: 1. Computing Systems 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet 5. Impacts of Computing
10201	Web Page Design	Web Page Design courses teach students how to design websites by introducing them to and refining their knowledge of site planning, page layout, graphic design, and the use of markup languages—such as Extensible Hypertext Markup, JavaScript, Dynamic HTML, Document Object Model, and Cascading Style Sheets—to develop and maintain a web page. These courses may also cover security and privacy issues, copyright infringement, trademarks, and other legal issues relating to the use of the Internet. Advanced topics may include the use of forms and scripts for database access, transfer methods, and networking fundamentals.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming May include these areas: 5. Impacts of Computing
10203	Interactive Media	Interactive Media courses provide students with the knowledge and skills to create, design, and produce interactive digital media products and services. The courses may emphasize the development of digitally generated and/or computer-enhanced media. Course topics may include 3D animation, graphic media, web development, and virtual reality. Upon completion of these courses, students may be prepared for industry certification.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming May include these areas: 5. Impacts of Computing

Course Code	Course Name	Description	Computer Science Standards
10205	Computer Gaming and Design	Computer Gaming and Design courses prepare students to design computer games by studying design, animation, artistic concepts, digital imaging, coding, scripting, multimedia production, and game play strategies. Advanced course topics include, but are not limited to, level design, environment and 3D modeling, scene and set design, motion capture, and texture mapping.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming May include these areas: 5. Impacts of Computing
10206	Mobile Applications	Mobile Applications courses provide students with opportunities to create applications for mobile devices using a variety of commercial and open source software. These courses typically address the installation and modification of these applications, as well as customer service skills to handle user issues.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming May include these areas: 5. Impacts of Computing
10251	Computer Technology	Computer Technology courses introduce students to the features, functions, and design of computer hardware and provide instruction in the maintenance and repair of computer components and peripheral devices.	CS topics include subjects and standards in these core areas: 1. Computing Systems 3. Data & Analysis 4. Algorithms & Programming May include these areas: 2. Networks & the Internet 5. Impacts of Computing
10253	Information Support and Services	Information Support and Services courses prepare students to assist users of personal computers by diagnosing their problems in using application software packages and maintaining security requirements.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 4. Algorithms & Programming May include these areas: 3. Data & Analysis 5. Impacts of Computing

Course Code	Course Name	Description	Computer Science Standards
10254	IT Essentials: PC Hardware and Software	IT Essentials: PC Hardware and Software courses provide students with in-depth exposure to computer hardware and operating systems. Course topics include the functionality of hardware and software components as well as suggested best practices in maintenance and safety issues. Students learn to assemble and configure a computer, install operating systems and software, and troubleshoot hardware and software problems. In addition, these courses introduce students to networking and often prepare them for industry certification.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 4. Algorithms & Programming May include these areas: 3. Data & Analysis 5. Impacts of Computing
10297	Information Support and Services—Independent Study	Information Support and Services—Independent Study courses, often conducted with instructors as mentors, enable students to explore topics related to computer information support and services. Independent Study courses may serve as an opportunity for students to expand their expertise in a particular specialization, to explore a topic in greater detail, or to develop more advanced skills.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 4. Algorithms & Programming May include these areas: 3. Data & Analysis 5. Impacts of Computing
10298	Information Support and Services—Workplace Experience	Information Support and Services—Workplace Experience courses provide students with work experience in fields related to information support and service. Goals are typically set cooperatively by the student, teacher, and employer (although students are not necessarily paid). These courses may include classroom activities as well, involving further study of the field or discussion regarding experiences that students encounter in the workplace.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 4. Algorithms & Programming May include these areas: 3. Data & Analysis 5. Impacts of Computing
10301	***Computer Forensics**	Computer Forensics courses address the preservation, identification, extraction, documentation, and interpretation of computer data. Topics covered may include legal concepts, evidence handling and preservation, file system structures, chain of custody, and identification and recovery of computer data. These courses may also cover the need to perform an investigation and how to collect evidence and analyze data.	CS topics include subjects and standards in these core areas: 1. Computing Systems 2. Networks & the Internet 3. Data & Analysis 4. Algorithms & Programming 5. Impacts of Computing

STANDARDS AND PRACTICES BY GRADE BAND GUIDE

Introduction

Within the Washington State Learning Standards for Computer Science (CS) reside a description of CS practices that translates the standards into teacher classroom instruction. The practices are vertically aligned, building on foundational knowledge in kindergarten and becoming increasingly complex as students increase their depth of understanding. As students move through each grade band, they move from novice upward realizing improved skills and abilities. This movement upward is called a learning or practice progression. Computer science practices are listed in the table below.

Table 5: Practice Statements

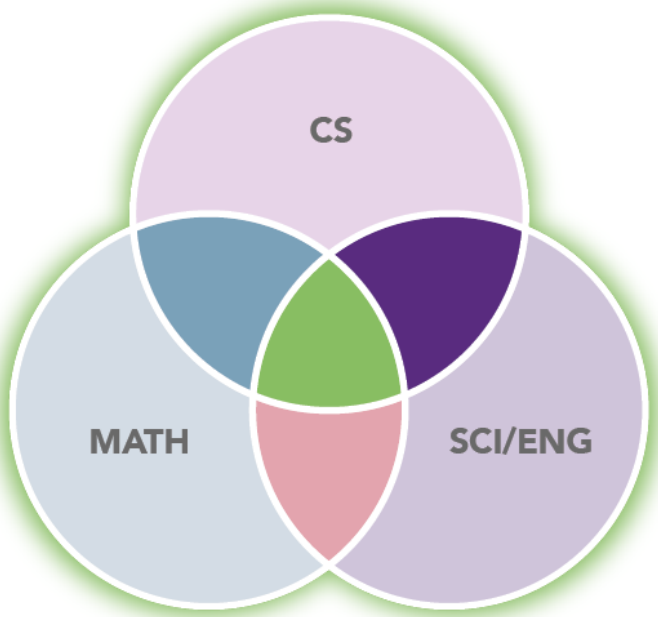
Practice	Practice Statements
1. Fostering and Inclusive Computing Culture	1.1 Include the unique perspectives of others and reflect on one's own perspectives when designing and developing computational products. 1.2 Address the needs of diverse end users during the design process to produce artifacts with broad accessibility and usability. 1.3 Employ self- and peer-advocacy to address bias in interactions, product design, and development methods.
2. Collaborating Around Computing	2.1 Cultivate working relationships with individuals possessing diverse perspectives, skills, and personalities. 2.2 Create team norms, expectations, and equitable workloads to increase efficiency and effectiveness. 2.3 Solicit and incorporate feedback from and provide constructive feedback to team members and other stakeholders. 2.4 Evaluate and select technological tools that can be used to collaborate on a project.
3. Recognizing and Defining Computational Problems	3.1 Identify complex, interdisciplinary, real-world problems that can be solved computationally. 3.2 Decompose complex real-world problems into manageable subproblems that could integrate existing solutions or procedures. 3.3 Evaluate whether it is appropriate and feasible to solve a problem computationally.
4. Developing and Using Abstractions	4.1 Extract common features from a set of interrelated processes or complex phenomena. 4.2 Evaluate existing technological functionalities and incorporate them into new designs. 4.3 Create modules and develop points of interaction that can apply to multiple situations and reduce complexity. 4.4 Model phenomena and processes and simulate systems to understand and evaluate potential outcomes.
Practice	**Practice Statements**

5. Creating Computational Artifacts	5.1 Plan the development of a computational artifact using an iterative process that includes reflection on and modification of the plan, taking into account key features, time and resource constraints, and user expectations. 5.2 Create a computational artifact for practical intent, personal expression, or to address a societal issue. 5.3 Modify an existing artifact to improve or customize it. Students should be able to examine existing artifacts to understand what they do.
6. Testing and Refining Computational Artifacts	6.1 Systematically test computational artifacts by considering all scenarios and using test cases. 6.2 Identify and fix errors using a systematic process. 6.3 Evaluate and refine a computational artifact multiple times to enhance its performance, reliability, usability, and accessibility.
7. Communicating About Computing	7.1 Select, organize and interpret large data sets from multiple sources to support a claim. 7.2 Describe, justify, and document computational processes and solutions using appropriate terminology consistent with the intended audience and purpose. 7.3 Articulate ideas responsibly by observing intellectual property rights and giving appropriate attribution.

The practices are grounded in the belief that computer science offers unique opportunities to apply to other subjects. Figure 3 describes the intersection among practices in computer science, science and engineering, and mathematics. Explicit instruction is required to create the connections illustrated in the figure.

Teachers across the state assisted in translating the CS standards into a familiar everyday language free from technical terms. This Description is included in the tables below to provide clarity to teachers and students. The practice progression illustrates what CS looks like when describing and engaging in activities in each grade band. Provided in the guidance are samples of student performance and additional sub-concept information within each grade level band.

RELATIONSHIPS BETWEEN COMPUTER SCIENCE, SCIENCE AND ENGINEERING, AND MATH PRACTICES



CS + Math

- **Develop and use abstractions**
  M2. Reason abstractly and quantitatively
  M7. Look for and make use of structure
  M8. Look for and express regularity in repeated reasoning
  CS4. Developing and Using Abstractions
- **Use tools when collaborating**
  M5. Use appropriate tools strategically
  CS2. Collaborating Around Computing
- **Communicate precisely**
  M6. Attend to precision
  CS7. Communicating About Computing

CS + Sci/Eng

- **Communicate with data**
  S4. Analyze and interpret data
  CS7. Communicating About Computing
- **Create artifacts**
  S3. Plan and carry out investigations
  S6. Construct explanations and design solutions
  CS4. Developing and Using Abstractions
  CS5. Creating Computational Artifacts
  CS6. Testing and Refining Computational Artifacts

CS + Math + Sci/Eng

- **Model**
  S2. Develop and use models
  M4. Model with mathematics
  CS4. Developing and Using Abstractions
  CS6. Testing and Refining Computational Artifacts
- **Use computational thinking**
  S5. Use mathematics and computational thinking
  CS3. Recognizing and Defining Computational Problems
  CS4. Developing and Using Abstractions
  CS5. Creating Computational Artifacts
- **Define problems**
  S1. Ask questions and define problems
  M1. Make sense of problems and persevere in solving them
  CS3. Recognizing and Defining Computational Problems
- **Communicate rationale**
  S7. Engage in argument from evidence
  S8. Obtain, evaluate, and communicate information
  M3. Construct viable arguments and critique the reasoning of others
  CS7. Communicating About Computing

* Computer science practices also overlap with practices in other domains, including English language arts. For example, *CS1. Fostering an Inclusive Computing Culture* and *CS2. Collaborating Around Computing* overlap with *E7. Come to understand other perspectives and cultures through reading, listening, and collaborations.*

Figure 3: Relationships between computer science, science and engineering, and math practices

K–2 STANDARDS & PRACTICES

Purpose: CS education for students in grades K–2 is focused on building a strong foundation of knowledge that supports continued CS learning, such as proper terminology, safety (passwords, internet), ethical responsibilities, and how computers are used to support and change our lives. Samples of student performance are provided for teachers and instructional staff to be better equipped to develop CS activities that align with the state standards.

1. Fostering an Inclusive Computing Culture

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 1	**1A-CS-01** Select and operate appropriate software to perform a variety of tasks and recognize that users have different needs and preferences for the technology they use.	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 need to complete.	**1.1** Students should begin to differentiate their technology preferences from the technology preferences of others.	• A student, when asked to draw a picture, should be able to open and use a drawing app/program to complete this task, or if asked to create a presentation, they should be able to open and use presentation software. • Students may compare different web browsers, or word processing, presentation, or drawing programs. • Students, with teacher guidance, should be able to compare and discuss preferences for software with the same primary functionality.	**Devices** People use computing devices to perform a variety of tasks accurately and quickly. Computing devices interpret and follow the instructions given literally.

2. Collaborating around Computing

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 2	**1A-IC-17** Work respectfully and responsibly with others online.	Online communication facilitates positive interactions, such as sharing ideas with many people., but the public and anonymous nature of online communication also allow intimidating and inappropriate behavior in the form of cyberbullying.	**2.1** Students 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.	• Students need to use their manners while using technology and while online. There should be no bullying, and they should use friendly language. • Students in the classroom will work on their devices, use appropriate manners, and will not bully. • Students will be helpful and cooperative to other students and adults and also treat the equipment with respect. • 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 with others. • Students could provide feedback to others about their work in a kind and respectful manner.	**Social Interactions** Computing has positively and negatively changed the way people communicate. People can have access to information and each other instantly, anywhere, and at any time, but they are at the risk of cyberbullying and reduced privacy.

3. Recognizing & Defining Computational Problems

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 3	**1A-AP-11** Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.	Decomposition is the act of breaking down tasks into simpler tasks.	**3.2** 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.	• Students could break down (or decompose) the steps needed to complete a task. • 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.	**Modularity** Complex tasks can be broken down into simpler instructions, some of which can be broken down even further. Likewise, instructions can be combined to accomplish complex tasks.

4. Developing & Using Abstractions

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**1A-DA-05** Store, copy, search, retrieve, modify, and delete information using a computing device and define the information stored as data.	All information stored and processed by a computing device referred to as data. Data can be images, text documents, audio files, software programs or apps, video files, etc.	**4.2** 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.	• Students use software to complete tasks on a computing device, and they will be manipulating data.	**Storage** Computers store data that can be retrieved later. Identical copies of data can be made and stored in multiple locations for a variety of reasons, such as to protect against loss.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**1A-DA-06** Collect and present the same data in various visual formats.	The collection and use of data about the world is a routine part of life and influences how people live. The data collected could then be organized into two or more visualizations, such as a bar graph, pie chart, or pictograph.	**4.4** Students with guidance may 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. **7.1** Students should, with guidance, present basic data using visual representations, such as storyboards, flowcharts, and graphs.	• Students may draw pictures of sunrises and sunsets, or show the stages in a process, such as brushing your teeth. • Students may create an animation to model a phenomenon, such as evaporation. • 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 throughout a storm. • Students could count the number of pieces of each color of candy in a bag of candy, using 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.	**Collection** Everyday digital devices collect and display data over time. The collection and use of data about individuals and the world around them is a routine part of life and influences how people live.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**1A-DA-07** Identify and describe patterns in data visualizations, such as charts or graphs, to make predictions.	Data can be used to make inferences or predictions about the world.	**4.1** Students should be able to identify and describe repeated sequences in data or code through analogy to visual patterns or physical sequences of objects.	• Students could use a number chart to figure out what comes next. • Students look at a pattern to play a game. • 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 present, and then predict 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 pattern observed.	**Visualization & Transformation** Data can be displayed for communication in many ways. People use computers to transform data into new forms, such as graphs and charts.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**1A-AP-08** Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.	An algorithm is the combination of smaller tasks into more complex tasks. Students could create and follow algorithms for making simple foods, brushing their teeth, getting ready for school, participating in clean-up time.	**4.4** Students with guidance 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.	• Students can make a list of steps to show how to complete a task Example: How to line up for lunch • Students can make a list of steps to show how to complete a task. It can be introduced in the first few weeks of school and practiced throughout the year. • Students have multiple routines that need to be followed; lining up, getting ready for class, walking to a meeting place, packing to go home. • Students can also create an animation to model a phenomenon, such as evaporation.	**Inference & Models** Data can be used to make inferences or predictions about the world. Inferences, statements about something that cannot be readily observed, are often based on observed data. Predictions, statements about future events, are based on patterns in data and can be made by looking at data visualizations, such as charts and graphs.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**1A-AP-09** Model the way programs store and manipulate data by using numbers or other symbols to represent information.	Information in the real world can be represented in computer programs.	**4.4** Students with guidance 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.	• Students create steps on a map to find their treasure, using arrows to tell which direction to walk and how many steps to take. • 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. • Students may also create an animation to model a phenomenon, such as evaporation.	**Variables** Information in the real world can be represented in computer programs. Programs store and manipulate data, such as numbers, words, colors, and images. The type of data determines the actions and attributes associated with it.

5. Creating Computational Artifacts

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 5	**1A-AP-10** Develop programs with sequences and simple loops, to express ideas or address a problem.	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.	**5.2** Students should be able to choose from a set of given commands to create simple animated stories or solve pre-existing problems.	• Students pick which steps will help their robot get to the end of a puzzle. • Students decide which comes first, second, third, and more to solve a task. • Students could correctly sequence a simple animated story. If the commands to program a robot are not in the correct order, the robot will not complete the task desired. • Students can use Loops allow for the repetition of a sequence of code multiple times. For example, in a program to show the life cycle of a butterfly, a loop could be combined with move commands to allow continual but controlled movement of the character	**Control** Computers follow precise sequences of instructions that automate tasks. Program execution can also be non-sequential by repeating patterns of instructions and using events to initiate instructions.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 5	**1A-AP-12** Develop plans that describe a program's sequence of events, goals, and expected outcomes.	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.	**5.1** Students, with the help of teachers, should participate in project planning and the creation of brainstorming documents. **7.2** Students 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).	• Students, with the help of their teacher, create a plan to solve a simple task. They work on the steps for making it work. For example, how to make a peanut butter and jelly sandwich. • Students practice using computer science vocabulary with their plans. • 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.	**Program Development** People develop programs collaboratively and for a purpose, such as expressing ideas or addressing problems.

6. Testing and Refining Computational Artifacts

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 6	**1A-AP-14** Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.	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.	**6.2** Students could use trial and error to fix simple errors. For example, a student may try reordering the sequence of commands in a program.	• Students could try reordering the sequence of commands in a program. • Students find and identify patterns, sequences, and repletion in a set of commands. They can then use these patterns to help find and fix mistakes in their commands. • Students practice debugging by following instructions and identify when there is a roadblock. They then use the identified pattern to fix the mistake. • Students in a hardware context could try to fix a device by resetting it or checking whether it is connected to a network	**Program Development** People develop programs collaboratively and for a purpose, such as expressing ideas or addressing problems.

7. Communicating About Computing

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**1A-CS-02** Use appropriate terminology in identifying and describing the function of common physical components of computing systems (hardware).	A computing system is composed of hardware and software. The hardware consists of physical components. 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.	**7.2** Students 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), and expected outcomes of their solutions.	• Students in kindergarten could be asked to: "Show me what you use to type the word "and," or "Show me what you use to hear a sound." • Students in first grade could be asked to: "Tell me and show me how to type the word "and," or "Tell me and show me how to hear the sound." • Students in second grade may be asked to: "1. Tell me how to type the "and," or "Tell me how to hear the sound.	**Hardware & Software** A computing system is composed of hardware and software. The hardware consists of physical components, while the software provides instructions for the system. These instructions are represented in a form that a computer can understand.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**1A-CS-03** Describe basic hardware and software problems using accurate terminology.	Problems with computing systems have different causes. Students at this level do not need to understand those causes. Still, 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.).	**6.2** Students could use trial and error to fix simple errors. For example, a student may try reordering the sequence of commands in a program. **7.2** Students 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).	• Students use basic computer terms to describe problems they may have with hardware and software. • 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.	**Troubleshooting** Computing systems might not work as expected because of hardware or software problems. Describing a problem is the first step toward finding a solution.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**1A-NI-04** Explain what passwords are and why we use them, and use strong passwords to protect devices and information from unauthorized access.	Learning to protect one's device or information from unwanted use by others is an essential first step in learning about cybersecurity.	**7.3** Students should apply these concepts to computational ideas and creations.	• Students are not required to use multiple strong passwords. They should appropriately use and protect the passwords they are required to use.	**Cybersecurity** Connecting devices to a network or the Internet provides many benefits. Still, care must be taken to use authentication measures, such as strong passwords, to protect devices and information from unauthorized access.
Practice 7	**1A-AP-13** Give attribution when using the ideas and creations of others while developing programs	Using computers comes with a level of responsibility. Proper attribution at this stage does not require a formal citation, such as in a bibliography or works cited document.	**7.3** Students should apply these concepts to computational ideas and creations.	• Students could credit artifacts that were created by others, such as pictures, music, and code. Credit could be given if presenting their work orally to the class, or in writing, if sharing work on a class blog or website.	**Program Development** People develop programs collaboratively and for a purpose, such as expressing ideas or addressing problems.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**1A-AP-15** Using correct terminology, describe steps are taken and choices made during the iterative process of program development.	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.	**7.2** Students 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).	• Students use computer science vocabulary when writing or saying what steps they took. For example, I wrote a "program" to find pictures in my story. • Students could describe outcomes by using coding journals, discussions with a teacher, class presentations, or blogs.	**Program Development** People develop programs collaboratively and for a purpose, such as expressing ideas or addressing problems.
Practice 7	**1A-IC-16** Compare how people live and work before and after the implementation or adoption of new computing technology.	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.	**7** Students with guidance, present basic data through the use of visual representations, using language that articulates what they are doing and identifies devices and concepts they are using with correct terminology while applying these concepts to computational ideas and creations.	• Students look at the technology that was around ten years ago and try to find current technology to compares. • Students hear stories about how people used to live. • Students create and share stories about what it may look like in the future. • Students may view and read the information on the Internet about a topic, or they can download e-books about it directly to a device.	**Culture** Computing technology has positively and negatively changed the way people live and work. Computing devices can be used for entertainment and as productivity tools, and they can affect relationships and lifestyles.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**1A-IC-18** Keep login information private, and log off devices appropriately	People use computing technology in ways that can help or hurt themselves or others. Harmful behaviors, such as sharing private information and leaving public devices logged in, should be recognized and avoided.	**7.3** Students should apply these concepts to computational ideas and creations.	• Students should know not to share their username and password with other people. • Students, when finishing using a computer, should always logoff. Logging off will ensure people who use the computer after you are not able to see your documents. • Students show the teacher they have logged off the computer. • Students should know not to share usernames and passwords, and teachers can keep the information until students memorize it.	**Social Interactions** Computing has positively and negatively changed the way people communicate. People can have access to information and each other instantly, anywhere, and at any time, but they are at the risk of cyberbullying and reduced privacy

GRADES 3–5 CS STANDARDS & PRACTICES

Purpose: CS education for students in grades 3–5 is focused on the continued expansion and deepening of foundational knowledge, such as learning how hardware and software work in tandem to move data, communicating with data, and the added value of diverse perspectives. Examples of CS in 3rd–5th grade classrooms include using Excel to organize and create visual data representations to answer questions. Samples of student performance are provided for teachers and instructional staff to be better equipped to develop CS activities and courses that meet the state requirements.

1. Fostering an Inclusive Computing Culture

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 1	**1B-AP-13** Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences.	Planning is an essential part of the iterative process of program development.	**1.1** Students should be presented with perspectives from people with different backgrounds, ability levels, and points of view. **5.1** Students with scaffolding should gain greater independence and sophistication in the planning, design, and evaluation of artifacts	• Students outline key features, time and resource constraints, and user expectations. • Students could document the plan as, for example, a storyboard, flowchart, pseudocode, or story map.	**Program Development** People develop programs using an iterative process involving design, implementation, and review. Design often involves reusing existing code or remixing other programs within a community. People continuously review whether programs work as expected, and they fix or debug, parts that do not. Repeating these steps enables people to refine and improve programs

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 1	**1B-IC-19** Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users.	The development and modification of computing technology are driven by people's needs and wants and can affect groups differently. Anticipating the needs and wants of diverse end-users requires students to purposefully consider potential perspectives of users with different backgrounds, ability levels, points of view, and disabilities.	**1.2** Students should consider the preferences of people who might use their products.	• Students may consider using both speech and text when they wish to convey information in a game. • Students may also wish to vary the types of programs they create, knowing that not everyone shares their tastes.	**Culture** The development and modification of computing technology are driven by people's needs and wants and can affect groups differently. Cultural practices can influence computing technologies
Practice 1	**1B-IC-20** Seek diverse perspectives for the purpose of improving computational artifacts.	Computing provides the possibility for collaboration and sharing of ideas and allows the benefit of diverse Perspectives, another grade level, forums, or website comments,	**1.1** Students should be presented with perspectives from people with different backgrounds, ability levels, and points of view.	• Students could seek feedback from other groups in their class or students from multiple sources for the purpose of improving computational artifacts • Students, with guidance from their teacher, could use video conferencing tools or other online collaborative spaces, such as blogs, wikis, to gather feedback from individuals and groups about programming projects.	**Social Interactions** Computing technology allows for local and global collaboration. By facilitating communication and innovation, computing influences many social institutions such as family, education, religion, and the economy.

2. Collaborating Around Computing

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 2	**1B-AP-16** Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development.	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.	**2.2** Students take on various roles, with the teacher guiding 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.	• Students take turns in different roles during program development, such as note-taker, facilitator, program tester, or "driver" of the computer.	**Program Development** People develop programs using an iterative process involving design, implementation, and review. Design often involves reusing existing code or remixing other programs within a community. People continuously review whether programs work as expected, and they fix or debug, parts that do not. Repeating these steps enables people to refine and improve programs

3. Recognizing & Defining Computational Problems

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 3	**1B-NI-05** Discuss real-world cybersecurity problems and how personal information can be protected.	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 digital.	**3.1** Students should ask clarifying questions to understand whether a problem or part of a problem can be solved using a computational approach.	• Students could discuss or use a journaling or blogging activity to explain, orally, or in writing about topics that relate to personal cybersecurity issues. • Students could choose discussion topics based on current events related to cybersecurity or topics that apply to students, such as: why is it necessary to back 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.	**Cybersecurity** Information can be protected using various security measures. These measures can be physical and digital.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 3	**1B-AP-11** Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.	Decomposition is the act of breaking down tasks into simpler tasks.	**3.2** Students 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 could decompose (break down) problems into smaller, manageable subproblems to facilitate the program (coding or problem-solving in other topics) development process. • Students could create an animation by separating a story into different scenes. For each scene, they would select a background, place characters, and program actions.	**Modularity** Programs can be broken down into smaller parts to facilitate their design, implementation, and review. Programs can also be created by incorporating smaller portions of programs that have been created.
Practice 3	**1B-IC-18** Discuss computing technologies that have changed the world, and express how those technologies influence, and are influenced by, cultural practices.	New computing technology and existing technologies are modified for many reasons, including to increase their benefits, decrease their risks, and meet societal needs.	**3.1** Students should ask clarifying questions to understand whether a problem or part of a problem can be solved using a computational approach.	• Students, with teacher guidance, 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, or how social media has influenced social and political changes.	**Culture** The development and modification of computing technology are driven by people's needs and wants and can affect groups differently. Computing technologies influence and are influenced by cultural practices.

4. Developing & Using Abstractions

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**1B-CS-02** Model how computer hardware and software work together as a system to accomplish tasks.	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.	**4.4** 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 spread.	• Students, with scaffolded support, can use the correct vocabulary to talk about parts of a device that they can see (i.e., headphones, headphone jack, mouse, keyboard, power, on/off buttons, etc.). • 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.	**Hardware and Software** Hardware and software work together as a system to accomplish tasks, such as sending, receiving, processing, and storing units of information as bits. Bits serve as the basic unit of data in computing systems and can represent a variety of information.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**1B-NI-04** Model how information is broken down into smaller pieces, transmitted as packets through multiple devices over networks and the Internet, and reassembled at the destination.	Information is sent and received over physical or wireless paths, broken down into smaller pieces called packets, which are sent independently and reassembled at the destination.	**4.4** 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 spread.	• Students can demonstrate their understanding of this flow of information by drawing a model of the way packets are transmitted or 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.	**Network Communication & Organization** Information needs a physical or wireless path to travel to be sent and received, and some paths are better than others. Information is broken into smaller pieces, called packets that are sent independently and reassembled at the destination. Routers and switches accurately send packets across paths to their destinations.

5. Creating Computational Artifacts

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 5	**1B-AP-09** Create programs that use variables to store and modify data.	Variables are used to store and modify data. At this level, understanding how to use variables is sufficient.	**5.2** Students should focus on artifacts of personal importance.	• Students may use mathematical operations to add to the score of a game or subtract from the number of lives available in a game. • Students can use a variable as a countdown timer is another example.	**Variables** Programming languages provide variables, which are used to store and modify data. The data type determines the values and operations that can be performed on that data.
Practice 5	**1B-AP-10** Create programs that include sequences, events, loops, and conditionals.	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.	**5.2** Students should focus on artifacts of personal importance.	• Students could write a program to explain the water cycle, and when a specific component is clicked (event), the program will show information about that part of the water cycle. Conditionals allow for the execution of a portion of code in a program when a specific condition is true. • Students could write a math game that asks multiplication fact questions and then uses a conditional to check whether the answer that was entered is correct. Loops allow for the repetition of a sequence of code multiple times. • Students can 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.	**Control** Control structures, including loops, event handlers, and conditionals, are used to specify the flow of execution. Conditionals selectively execute or skip instructions under different conditions.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 5	**1B-AP-12** Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.	Programs can be broken down into smaller parts, which can be incorporated into new or existing programs.	**5.3** Students should attempt to use existing solutions to accomplish the desired goal. For example, students could attach a programmable light sensor to a physical artifact they have created to make it respond to light.	• Give credit where credit is due. Observe intellectual property rights and give appropriate attribution when creating or remixing programs. • Students could modify prewritten code from a single-player game to create a two-player game: with slightly different rules; remix and add another scene to an animated story; use code to make a ball bounce from another program in a new basketball game, or modify an image created by another student.	**Modularity** Programs can be broken down into smaller parts to facilitate their design, implementation, and review. Programs can also be created by incorporating smaller portions of programs that have already been created.

6. Testing and Refining Computational Artifacts

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 6	**1B-CS-03** Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies.	Although computing systems may vary, common troubleshooting strategies can be used on all of them.	**6.2** Students could use trial and error to fix simple errors. For example, a student may try reordering the sequence of commands in a program	• 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. • Students could solve errors that occur at school by using various strategies: such as rebooting the device; checking for power; checking network availability; closing and reopening an app; making sure to turn speakers on; headphones are plugged in, and making sure that the caps lock key is not on, to solve these problems, when possible.	**Troubleshooting** Computing systems share similarities, such as the use of power, data, and memory. Common troubleshooting strategies, such as checking that power is available, checking that physical and wireless connections are working, and clearing out the working memory by restarting programs or devices, are effective for many systems.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 6	**1B-AP-08** Compare and refine multiple algorithms for the same task and determine which is the most appropriate.	Different algorithms can achieve the same result, though sometimes one algorithm might be most appropriate for a specific situation.	**6.3** 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.	• Students should be able to look at different ways to solve the same task and decide which would be the best solution. • 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.	**Algorithms** Different algorithms can achieve the same result. Some algorithms are more appropriate for a specific context than others.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 6	**1B-AP-15** Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.	As students develop programs, they should continuously test those programs to see that they do what was expected and fix (debug), any errors.	**6.1** Students should test computational artifacts by considering potential errors, such as what will happen if a user enters invalid input. **6.2** Students could use trial and error to fix simple errors. For example, a student may try reordering the sequence of commands in a program	• Students should be able to debug simple errors in programs created by others successfully. • Students, in a hardware context, could try to fix a device by resetting it or checking whether it is connected to a network	**Program Development** People develop programs using an iterative process involving design, implementation, and review. Design often involves reusing existing code or remixing other programs within a community. People continuously review whether programs work as expected, and they fix or debug, parts that do not. Repeating these steps enables people to refine and improve programs.

7. Communicating About Computing

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**1B-CS-01** Describe how internal and external parts of computing devices function to form a system.	Computing devices often depend on other devices or components. Students should be able to describe how devices and components interact using correct terminology.	**7.2** Students should identify the goals and expected outcomes of their solutions	• Students describe how internal and external parts of computing devices function (Input, Process, Store, Output) to form a system. 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. • Students could describe how keyboard input, or a mouse click could cause an action to happen or information displayed on a screen; this could only happen because the computer has a processor to evaluate what is happening externally and produce corresponding responses.	**Devices** Computing devices may be connected to other devices or components to extend their capabilities, such as sensing and sending information. Connections can take many forms, such as physical or wireless. Together, devices and components form a system of interdependent parts that interact for a common purpose.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**1B-DA-06** Organize and present collected data visually to highlight relationships and support a claim.	Raw data has little meaning on its own. Data sorted or grouped to provide additional clarity. Organizing data can make interpreting and communicating it to others more accessible. The same data could be manipulated in different ways to emphasize aspects or parts of the data set.	**7.1** Students should, with guidance, present basic data using visual representations, such as storyboards, flowcharts, and graphs.	• Students use technology to organize and present collected data visually to show relationships and support a claim. • Students use data to tell or view a process, supporting an idea visually. It can be expanded to say: why a result happens, acknowledging the owner of the idea or knowledge. • Students may use a data set of sports teams that 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.	**Collection** People select digital tools for the collection of data based on what is being observed and how the data will be used. For example, a digital thermometer is used to measure temperature, and a GPS sensor is used to track locations. **Visualization & Transformation** People select aspects and subsets of data to be transformed, organized, clustered and categorized to provide different views and communicate insights gained from the data.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**1B-DA-07** Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea.	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.	**7.1** Students should, with guidance, present basic data with visual representations, such as storyboards, flowcharts, and graphs.	• Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea. • Students could refer to data when communicating an idea. For example, to explore the relationship between speed, time, and distance, students could operate a robot at a uniform speed, and at increasing time intervals to predict how far the robot travels at that speed. • 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 communicated could be inaccurate.	**Inference & Models** The accuracy of inferences and predictions is related to how realistically data is represented. Many factors influence the accuracy of inferences and predictions, such as the amount and relevance of data collected.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**1B-AP-14** Observe intellectual property rights and give appropriate attribution when creating or remixing programs.	Intellectual property rights can vary by country, but copyright laws give the creator of a work a set of rights that prevents others from copying the work and using it in ways that they may not like.	**5.2** Students should focus on artifacts of personal importance. **7.3** Students should identify instances of remixing, when ideas are borrowed and iterated upon, and give proper attribution.	• Students could 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. • Students' attributions should be written in the format required by the teacher and should always be included in any programs shared online.	**Program Development** People develop programs using an iterative process involving design, implementation, and review. Design often involves reusing existing code or remixing other programs within a community. People continuously review whether programs work as expected, and they fix or debug, parts that do not. Repeating these steps enables people to refine and improve programs.
Practice 7	**1B-AP-17** Describe choices made during program development using code comments, presentations, and demonstrations.	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.	**7.2** Students should identify the goals and expected outcomes of their solutions.	• These explanations could manifest themselves as in-line code comments for collaborators and assessors, or as parts of a summative presentation, such as a code walk-through or coding journal.	**Program Development** People develop programs using an iterative process involving design, implementation, and review. Design often involves reusing existing code or remixing other programs within a community. People continuously review whether programs work as expected, and they fix or debug, parts that do not.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**1B-IC-21** Use public domain or creative commons media, and refrain from copying or using material created by others without permission.	Ethical complications arise from the opportunities provided by computing. The ease of sending and receiving copies of media on the Internet, such as video, photos, and music, creates the opportunity for unauthorized use, such as online piracy, and disregard of copyrights.	**7.3** Students should identify instances of remixing, when ideas are borrowed and iterated upon, and give proper attribution.	• 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.	**Safety Law & Ethics** Ethical complications arise from the opportunities provided by computing. The ease of sending and receiving copies of media on the Internet, such as video, photos, and music, creates the opportunity for unauthorized use, such as online piracy, and disregard of copyrights, such as lack of attribution.

GRADES 6–8 STANDARDS & PRACTICES

Purpose: CS education for students in grades 6–8 is focused on gaining a more in-depth understanding that computers simply execute instructions, that the quality of data varies, and that data can be misrepresented. Samples of student performance are provided for teachers and instructional staff to be better equipped to develop CS activities and courses that meet the state requirements.

1. Fostering an Inclusive Computing Culture

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 1	**2-IC-21** Discuss issues of bias and accessibility in the design of existing technologies	Students should test and discuss the usability of various technology tools (e.g., apps, games, and devices) with the teacher's guidance.	**1.2** Students should be able to evaluate the accessibility of a product to a broad group of end-users, such as people with various disabilities.	• Students can discuss issues of bias and accessibility in the design of existing technologies. • Students can discuss facial recognition software that works better for lighter skin tones was likely developed with a homogeneous testing group and could improve by sampling a more diverse population. • Students discussing accessibility 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	**Culture** Advancements in computing technology change people's everyday activities. Society is faced with tradeoffs due to the increasing globalization and automation that computing brings.

2. Collaborating Around Computing

#	Standards	Accessible language	Practice Progression	Samples of student performance	Sub-Concept
Practice 2	**2-DA-07** Represent data using multiple encoding schemes.	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).	**4** Students should extract common features from more complex phenomena or processes. Students should begin to 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). Within an object-oriented programming context, module design may include defining the interactions among objects, combining both data and procedures, and documented for reuse in other programs.	• Students could represent the same data in multiple ways, using different methods. • 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). • 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. • Students could be able to design systems of interacting modules, each with a well-defined role that coordinates to accomplish a common goal.	**Storage** Applications store data as a representation. Representations occur at multiple levels, from the arrangement of information into organized formats (such as tables in software) to the physical storage of bits. The software tools used to access information translate the low-level representation of bits into a form understandable by people.

#	Standards	Accessible language	Practice Progression	Samples of student performance	Sub-Concept
Practice 2	**2-AP-15** Seek and incorporate feedback from team members and users to refine a solution that meets user needs.	Development teams that employ user-centered design create solutions (e.g., programs and devices) that can have a significant societal impact, such as an app that allows people with speech difficulties to translate hard-to-understand pronunciation into understandable language.	**2.3** 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.	• Students should begin to seek diverse perspectives throughout the design process to improve their computational artifacts. • Students should consider the end-user may include usability, accessibility, age-appropriate content, respectful language, user perspective, pronoun use, color contrast, and ease of use.	**Program Development** People design meaningful solutions for others by defining a problem's criteria and constraints, carefully considering the diverse needs and wants of the community and testing whether criteria and constraints are met**.**
Practice 2	**2-AP-18** Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.	Collaboration is a common and crucial practice in programming development. Often, many individuals and groups work on the interdependent parts of a project together.	**2.2** Students 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.	• Students could 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. • Students may divide the design stage of a game into planning the storyboard, flowchart, and different parts of the game mechanics. They can then assign deadlines.	**Program Development** People design meaningful solutions for others by defining a problem's criteria and constraints, carefully considering the diverse needs and wants of the community and testing whether criteria and constraints were met.

#	Standards	Accessible language	Practice Progression	Samples of student performance	Sub-Concept
Practice 2	**2-IC-22** Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.	Crowdsourcing is gathering services, ideas, or content from a large group of people, especially from the online community. It can be done at the local level (e.g., classroom or school) or global level (e.g., age-appropriate online communities, like Scratch and Minecraft).	**2.4** Students should also begin to make decisions about which tools would be best to use and when to use them. **5.2** Students expressions should become more complex and of increasingly broader significance	• Student groups could combine animations to create a digital community mosaic. They could also solicit feedback from many people through the use of online communities and electronic surveys.	**Social Interactions** People can organize and engage around issues and topics of interest through various communication platforms enabled by computing, such as social networks and media outlets. These interactions allow issues to be examined using multiple viewpoints from a diverse audience.

3. Recognizing & Defining Computational Problems

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 3	**2-CS-01** Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.	The study of human-computer interaction (HCI) can improve the design of devices, including both hardware and software. Including consideration of usability through several lenses, including accessibility, ergonomics, and learnability.	**3.3** Students should systematically evaluate the feasibility of using computational tools to solve given problems or subproblems, such as through a cost-benefit analysis.	• Students could make recommendations for existing devices (e.g., a laptop, phone, or tablet) or design components or interface (e.g., create controllers) by considering usability through several lenses, including accessibility, ergonomics, and learnability • Students can research, assistive devices capabilities such as scanning written information and converting it to speech.	**Devices** The interaction between humans and computing devices presents advantages, disadvantages, and unintended consequences. The study of human-computer interaction can improve the design of devices and extend the abilities of humans.
Practice 3	**2-AP-13** Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.	Students should break down problems into subproblems, which can be further broken down into smaller parts.	**3.2** Students should break down a program into sub-goals getting input from the user, processing the data, and displaying the result to the user.	• Students as part of program development focus on one piece at a time (e.g., getting input from the user, processing the data, and displaying the result to the user). • Students can work on different parts of a problem at the same time, and animations can be decomposed into multiple scenes, which can be independently developed.	**Modularity** Programs use procedures to organize code, hide implementation details, and make code easier to reuse. Procedures can be repurposed in new programs. Defining parameters for procedures can generalize behavior and increase reusability.

4. Developing and Using Abstractions

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**2-AP-14** Create procedures with parameters to organize code and make it easier to reuse.	Students should create procedures and 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.	**4.1** Students should extract common features from more complex phenomena or processes. **4.3** Students should be able to design systems of interacting modules, each with a well-defined role that coordinates to accomplish a common goal.	• Students can create a procedure to draw a circle, which involves many instructions, but all of them can be invoked with one instruction, such as "drawCircle." By adding a radius parameter, the user can easily draw circles of different sizes. • 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. • Students may design module, with objects that include defining the interactions among objects. At this stage, these modules, which combine both data and procedures, can be designed and documented for reuse in other programs.	**Modularity** Programs use procedures to organize code, hide implementation details, and make code easier to reuse. Procedures can be repurposed in new programs. Defining parameters for procedures can generalize behavior and increase reusability**.**

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**2-NI-04** Model the role of protocols in transmitting data across networks and the Internet.	Protocols are rules that define how messages are sent between computers. They determine how quickly and securely information is transmitted across networks and the Internet, how to handle errors in transmission. 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.	**4.4** 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.	• Students could model how data is sent using protocols to choose the fastest path, to deal with missing information, and to deliver sensitive data • Students could devise a plan for resending lost information or for interpreting a picture that has missing pieces. • Students might create a simple producer-consumer ecosystem model using a programming tool.	**Network Communication & Organization** Computers send and receive information based on a set of rules called protocols. Protocols define how messages between computers are structured and sent. Considerations of security, speed, and reliability are used to determine the best path to send and receive data.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**2-NI-06** Apply multiple methods of encryption to model the secure transmission of information.	Encryption can be as simple as letter substitution or as complicated as modern methods used to secure networks and the Internet.	**4.4** 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.	• Students could 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. • Students could secure messages using methods such as Caesar ciphers or steganography (i.e., hiding messages inside a picture or other data). They can also model more complicated methods, such as public-key encryption, through unplugged activities. • Students might create a simple producer-consumer ecosystem model using a programming tool.	**Cybersecurity** The information sent and received across networks can be protected from unauthorized access and modification in a variety of ways, such as encryption to maintain its confidentiality and restricted access to maintain its integrity. Security measures to safeguard online information proactively address the threat of breaches to personal and private data.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**2-AP-10** Use flowcharts and/or pseudocode to address complex problems as algorithms.	Complex problems are problems that would be difficult for students to solve computationally. Students should use pseudocode and flowcharts to organize and sequence an algorithm that addresses a complex problem, even though they may not program the solutions.	**4.1** Students should extract common features from more complex phenomena or processes. **4.4** 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.	• 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. • Students might create a simple producer-consumer ecosystem model using a programming tool. • Students could 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.	**Algorithms** Algorithms affect how people interact with computers and the way computers respond. People design algorithms that are generalizable to many situations. Readable algorithms are more comfortable to follow, test, and debug.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**2-AP-16** Incorporate existing code, media, and libraries into original programs, and give attribution.	Building on the work of others enables students to produce more interesting and powerful creations. Students should give attribution to the original creators to acknowledge their contributions.	**4.2** 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). **7.3** Students should also recognize the contributions of collaborators.	• Students should use portions of code, algorithms, and/or digital media in their programs and websites. At this level, they may also import libraries and connect to web application program interfaces (APIs). • Students in creating a side-scrolling game 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.	**Program Development** People design meaningful solutions for others by defining a problem's criteria and constraints, carefully considering the diverse needs and wants of the community and testing whether criteria and constraints were met.

5. Creating Computational Artifacts

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 5	**2-CS-02** Design projects that combine hardware and software components to collect and exchange data.	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.	**5.1** 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 can use both hardware and software to design a project that shares data • Students can choose components for a mobile app that 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	**Hardware & Software** Hardware and software determine a computing system's capability to store and process information. The design or selection of a computing system involves multiple considerations and potential tradeoffs, such as functionality, cost, size, speed, accessibility, and aesthetics.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 5	**2-AP-12** Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.	Control structures can be combined in many ways. Nested loops are loops placed within loops. Compound conditionals combine two or more conditions in a logical relationship (e.g., using AND, OR, and NOT), and nesting conditionals within one another allows the result of one conditional on leading to another.	**5.1** 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. **5.2** Students' expressions should become more complex and of increasingly broader significance.	• Students can, when programming an interactive story, use a compound conditional within a loop to unlock a door only if a character has a key AND is touching the door.	**Control** Programmers select and combine control structures, such as loops, event handlers, and conditionals, to create more complex program behavior.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 5	**2-DA-09** Refine computational models based on the data they have generated.	A model may be a programmed simulation of events or a representation of how various data is related. 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.	**5.3** Students should attempt to use existing solutions to accomplish the desired goal. For example, students could attach a programmable light sensor to a physical artifact they have created to make it respond to light.	• Students may predict 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). • Students could refine game mechanics based on test outcomes to make the game more balanced or fair.	**Inference & Models** Computer models can be used to simulate events, examine theories and inferences, or make predictions with either a few or millions of data points. Computer models are abstractions that represent phenomena and use data and algorithms to emphasize key features and relationships within a system. As more data is automatically collected, models can be refined.
Practice 5	**2-AP-11** Create clearly named variables that represent different data types and perform operations on their values.	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.	**5.1** 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 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.	**Variables** Programmers create variables to store data values of selected types. A meaningful identifier is assigned to each variable to access and perform operations on the value by name. Variables enable the flexibility to represent different situations, process different sets of data, and produce different outputs.

6. Testing and Refining Computational Artifacts

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 6	**2-DA-08** Collect data using computational tools and transform the data to make it more useful and reliable.	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.	**6.3** Students' evaluation and refinement should become an iterative process that also encompasses making artifacts more usable and accessible. 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	• Students could transform data to remove errors, highlight or expose relationships, and make it easier for computers to process. • Students could clean data, which is an essential transformation for ensuring consistent format and reducing noise and errors (e.g., removing irrelevant responses in a survey). • Students can transform collected data into something meaningful, preferably visual. An example of a transformation that highlights a relationship is representing males and females as percentages of a whole.	**Collection Visualization & Transformation** Data can be transformed to remove errors, highlight, or expose relationships, and make it easier for computers to process.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 6	**2-CS-03** Systematically identify and fix problems with computing devices and their components.	Since a computing device may interact with interconnected devices within a system, problems may not be due to the specific computing device itself but to devices connected to it.	**6.2** 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 thinking.	• Students could use checklists to troubleshoot problems with aircraft systems or use a similar, structured process to troubleshoot problems with computing systems and ensure that potential solutions are not overlooked. • Students could use troubleshooting strategies to include following a troubleshooting flow diagram, making changes to the software to see if the hardware will work, checking connections and settings, and swapping in working components. • Students could step through their program, line by line, to identify a loop that does not terminate as expected.	**Troubleshooting** Comprehensive troubleshooting requires knowledge of how computing devices and components work and interact. A systematic process will identify the source of a problem, whether within a device or in a more extensive system of connected devices.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 6	**2-AP-17** Systematically test and refine programs using a range of test cases.	Use cases and test cases are created and analyzed to meet the needs of users better 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.	**6.1** Students testing should become a deliberate process that is more iterative, systematic, and proactive.	• 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).	**Program Development** People design meaningful solutions for others by defining a problem's criteria and constraints, carefully considering the diverse needs and wants of the community and testing whether criteria and constraints were met.

7. Communicating About Computing

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**2-IC-20** Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.	Advancements in computer technology are neither wholly positive nor negative. However, the ways that people use computing technologies have tradeoffs.	**7.2** 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.	• Students could describe the positive and negative impacts of computers that affect daily life 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.	**Culture** Advancements in computing technology change people's everyday activities. Society is faced with tradeoffs due to the increasing globalization and automation that computing brings.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**2-IC-23** Describe tradeoffs between allowing information to be public and keeping information private and secure.	Sharing information online can help establish, maintain, and strengthen connections between people.	**7.2** 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.	• Students can describe the importance of keeping your information secure. Example questions: What are some of the possible outcomes of having your data compromised? What are some methods for maintaining it securely? • Students could research artists and designers that 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.	**Safety Law & Ethics** There are tradeoffs between allowing information to be public and keeping information private and secure. People can be tricked into revealing personal information when more public information is available about them online.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**2-NI-05** Explain how physical and digital security measures protect electronic information.	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 back-up copies on external storage devices, and erasing a storage device before it is reused.	**7.2** 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.	• Students could research digital security measures, which 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.	**Cybersecurity** The information sent and received across networks can be protected from unauthorized access and modification in a variety of ways, such as encryption to maintain its confidentiality and restricted access to maintain its integrity. Security measures to safeguard online information proactively address the threat of breaches to personal and private data.

GRADES 9–10 STANDARDS & PRACTICES

Purpose: CS education for students in 9th and 10th grade is focused on deepening understanding of why and how computing technologies work and then to build upon that conceptual knowledge by creating computational artifacts. Samples of student performance are provided for teachers and instructional staff to be better equipped to develop CS activities and courses that meet the state requirements.

1. Fostering an Inclusive Computing Culture

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 1	**3A-IC-24** Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.	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.	**1.2** 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.	• Students could 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 could begin to identify potential bias during the design process to maximize accessibility in product design. • Students can test an app and recommend to its designers that they respond to verbal commands to accommodate users who are blind or have physical disabilities.	**Culture** The design and use of computing technologies and artifacts can improve, worsen, or maintain inequitable access to information and opportunities.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 1	**3A-IC-25** Test and refine computational artifacts to reduce bias and equity deficits.	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.	**1.2** 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.	• 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. • Students can test an app and recommend to its designers that they respond to verbal commands to accommodate users who are blind or have physical disabilities.	**Culture** The design and use of computing technologies and artifacts can improve, worsen, or maintain inequitable access to information and opportunities.

2. Collaborating Around Computing

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 2	**3A-AP-22** Design and develop computational artifacts working in team roles using collaborative tools.	Collaborative tools could be as complicated as the 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.	**2.4** 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.	• 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	**Program Development** Diverse teams can develop programs with a broad impact through careful review and by drawing on the strengths of members in different roles. Design decisions often involve tradeoffs. The development of complex programs is aided by resources such as libraries and tools to edit and manage parts of the program. Systematic analysis is critical for identifying the effects of lingering bugs.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 2	**3A-IC-27** Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields.	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 different career fields has changed the nature and content of many careers.	**2.4** 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.	• Students could 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. • Students could compare ways different social media tools could help a team become more cohesive	**Social Interactions** 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 different career fields has changed the nature and content of many careers.

3. Recognizing & Defining Computational Problems

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 3	**3A-NI-06** Recommend security measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.	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, back-ups, and other measures.	**3.3** Students should include more factors in their evaluations, such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns	• Students could systematically evaluate the feasibility of using computational tools to solve given problems or subproblems, such as through a cost-benefit analysis. • Students should begin to include more factors in their evaluations, such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns.	**Cybersecurity** 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.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 3	**3A-DA-10** Evaluate the tradeoffs in how data elements are organized and where data is stored.	People make choices about how data elements are organized and where data is stored. These choices affect cost, speed, reliability, accessibility, privacy, and integrity.	**3.3** Students should include more factors in their evaluations, such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns	• Students could evaluate whether a chosen solution is most appropriate for a 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	**Storage** Data can be composed of multiple data elements that relate to one another. For example, population data may contain information about age, gender, and height. People make choices about how data elements are organized and where data is stored. These choices affect cost, speed, reliability, accessibility, privacy, and integrity.
Practice 3	**3A-AP-17** Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.	At this level, students should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist.	**3.2** Students encounter complex real-world problems that span multiple disciplines or social systems, and they should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist.	• Students could create an app to solve a community problem by connecting to an online database through an application programming interface (API).	**Control** Programmers consider tradeoffs related to implementation, readability, and program performance when selecting and combining control structures.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 3	**3A-IC-26** Demonstrate ways a given algorithm applies to problems across disciplines.	Computation can share features with disciplines such as art and music by algorithmically translating human intention into an artifact.	**3.1** Students should be able to identify real-world problems that span multiple disciplines and can be solved computationally.	• Students could 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	**Culture** The design and use of computing technologies and artifacts can improve, worsen, or maintain inequitable access to information and opportunities.

4. Developing and Using Abstractions

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**3A-CS-01** Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.	Computing devices are often integrated with other systems, including biological, mechanical, and social systems. The creation of integrated or embedded systems is not an expectation at this level.	**4.1** Students, when working with data, should be able to identify important aspects and find patterns in related data sets such as crop output, fertilization methods, and climate conditions.	• Students might select an embedded device such as a car stereo: identify the types of data (radio station presets, volume level) and procedures (increase volume, store/recall saved station; mute) it includes and explain how the implementation details are hidden from the user. • Students might select a medical device that 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 specific frequencies and magnify others.	**Devices** Computing devices are often integrated with other systems, including biological, mechanical, and social systems. These devices can share data. The usability, dependability, security, and accessibility of these devices, and the systems they are integrated with, are essential considerations in their design as they evolve**.**

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**3A-CS-02** Compare levels of abstraction and interactions between application software, system software, and hardware layers.	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.	**4.1** Students, when working with data, should be able to identify important aspects and find patterns in related data sets such as crop output, fertilization methods, and climate conditions.	• Students could compare how 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. • Students could compare system software that is used on many different types of devices, such as smart TVs, assistive devices, virtual components, cloud components, and drones. • 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.	**Hardware & Software** Levels of interaction exist between the hardware, software, and user of a computing system. The most common levels of software that a user interacts with include system software and applications. System software controls the flow of information between hardware components used for input, output, storage, and processing.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**3A-NI-04** Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing.	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.	**4.1** Students, when working with data, should be able to identify important aspects and find patterns in related data sets such as crop output, fertilization methods, and climate conditions.	• Students could use online network simulators to experiment with these factors.	**Network Communication & Organization** Network topology is determined, in part, by how many devices can be supported. Each device is assigned an address that uniquely identifies it on the network. The hierarchy and redundancy enable the scalability and reliability of the Internet in networks.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**3A-DA-09** Translate between different bit representations of real-world phenomena, such as characters, numbers, and images.	Data is represented by binary, hexadecimal, hexadecimal color codes, decimal percentages, ASCII/Unicode representation, and logic gates	**4.1** Students, when working with data, should be able to identify important aspects and find patterns in related data sets such as crop output, fertilization methods, and climate conditions.	• Students could convert hexadecimal color codes to decimal percentages, ASCII/Unicode representation, and logic gates	**Storage** Data can be composed of multiple data elements that relate to one another. For example, population data may contain information about age, gender, and height. People make choices about how data elements are organized and where data is stored. These choices affect cost, speed, reliability, accessibility, privacy, and integrity.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**3A-DA-11** Create interactive data visualizations using software tools to help others better understand real-world phenomena.	People transform, generalize, simplify, and present large data sets in different ways to influence how other people interpret and understand the underlying information.	**4.4** Students 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.	• Students could 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. • Students could create software tools or programming to create powerful, interactive data visualizations and perform a range of mathematical operations to transform and analyze data. Examples include visualization, aggregation, rearrangement, and application of mathematical operations.	**Collection Visualization & Transformation** Data is continuously collected or generated through automated processes that are not always evident, raising privacy concerns. The different collection methods and tools that are used influence the amount and quality of the data that is observed and recorded. People transform, generalize, simplify, and present large data sets in different ways to influence how other people interpret and understand the underlying information.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 4	**3A-DA-12** Create computational models that represent the relationships among different elements of data collected from a phenomenon or process.	Computational models make predictions about processes or phenomena based on selected data and features.	**4.4** Students 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.	• Students could analyze and evaluate these models against real-world observations. For example, students might create a simple producer-consumer ecosystem model using a programming tool. • Students could model phenomena as systems, with rules governing the interactions within the system. • Students 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.	**Inference & Models** The accuracy of predictions or inferences depends upon the limitations of the computer model and the data the model is built. The amount, quality, and diversity of data and the features chosen can affect the quality of a model and the ability to understand a system. Predictions or inferences are tested to validate models.
Practice 4	**3A-AP-14** Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.	Data is stored in multiple ways that support the solution of computational problems.	**4.1** Students, when working with data, should be able to identify important aspects and find patterns in related data sets such as crop output, fertilization methods, and climate conditions.	• Students should be able to identify common features in multiple segments of code and substitute a single segment that uses lists (arrays) to account for the differences.	**Variables** Information in the real world can be represented in computer programs. Programs store and manipulate data, such as numbers, words, colors, and images. The type of data determines the actions and attributes associated with it.

5. Creating Computational Artifacts

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 5	**3A-AP-13** Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.	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.	**5.2** Students should engage in the independent, systematic use of design processes to create artifacts that solve problems with social significance by seeking input from broad audiences.	• Students create artifacts that are personally relevant or beneficial to their community and beyond. • Students could develop artifacts in response to a task or a computational problem that demonstrates the performance, reusability, and ease of implementation of an algorithm.	**Algorithms** People evaluate and select algorithms based on performance, reusability, and ease of implementation. Knowledge of common algorithms improves how people develop software, secure data, and store information.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 5	**3A-AP-15** Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.	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.	**5.2** Students should engage in the independent, systematic use of design processes to create artifacts that solve problems with social significance by seeking input from broad audiences.	• Students might compare the readability and program performance of iterative and recursive implementations of procedures that calculate the Fibonacci sequence.	**Control** Programmers consider tradeoffs related to implementation, readability, and program performance when selecting and combining control structures.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 5	**3A-AP-16** Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.	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.	**5.2** Students should engage in the independent, systematic use of design processes to create artifacts that solve problems with social significance by seeking input from broad audiences.	• At previous levels, students have learned to create and call procedures. Here, 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	**Control** Programmers consider tradeoffs related to implementation, readability, and program performance when selecting and combining control structures.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 5	**3A-AP-18** Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.	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. 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.	**5.2** Students should engage in the independent, systematic use of design processes to create artifacts that solve problems with social significance by seeking input from broad audiences.	• 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	**Modularity** Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program, combinations of data and procedures, or independent but interrelated programs. Modules allow for better management of complex tasks.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 5	**3A-AP-19** Systematically design and develop programs for broad audiences by incorporating feedback from users.	Examples of programs could include games, utilities, and mobile applications. Students at lower levels collect feedback and revise programs. At this level, students should do so through a systematic process that includes feedback from broad audiences.	**5.1** Students should be capable of reflecting on and, if necessary, modifying the plan to accommodate end goals.	• Students might create a user satisfaction survey and brainstorm distribution methods that could yield feedback from a diverse audience, documenting the process they took to incorporate selected feedback in product revisions.	**Modularity** Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program, combinations of data and procedures, or independent but interrelated programs. Modules allow for better management of complex tasks.

6. Testing and Refining Computational Artifacts

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 6	**3A-CS-03** Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.	Troubleshooting complex problems involves the use of multiple sources when researching, evaluating, and implementing potential solutions. Troubleshooting also relies on experience, such as when people recognize that a problem is like one that they have seen before or adapt solutions that have worked in the past.	**6.2** 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.	• Students develop complex troubleshooting strategies include resolving connectivity problems, adjusting system configurations and settings, ensuring hardware and software compatibility, and transferring data from one device to another. • Students could create a flow chart, a job aid for a help desk employee, or an expert system.	**Troubleshooting** Troubleshooting complex problems involves the use of multiple sources when researching, evaluating, and implementing potential solutions. Troubleshooting also relies on experience, such as when people recognize that a problem is similar to one that they have seen before or adapt solutions that have worked in the past.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 6	**3A-NI-07** Compare various security measures, considering tradeoffs between the usability and security of a computing system.	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.	**6.3** Student's 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.	• 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.	**Network Communication & Organization** Network topology is determined, in part, by how many devices can be supported. Each device is assigned an address that uniquely identifies it on the network. The hierarchy and redundancy enable the scalability and reliability of the Internet in networks.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 6	**3A-AP-21** Evaluate and refine computational artifacts to make them more usable and accessible.	Testing and refinement are 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.	**6.3** Students' 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.	• Students could respond to the changing needs and expectations of end-users and improve the performance, reliability, usability, and accessibility of artifacts. • 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.	**Program Development** Diverse teams can develop programs with a broad impact through careful review and by drawing on the strengths of members in different roles. Design decisions often involve tradeoffs, the development of complex programs aided by resources such as libraries and tools to edit and manage parts of the program. Systematic analysis is critical for identifying the effects of lingering bugs.

7. Communicating About Computing

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**3A-NI-05** Give examples to illustrate how sensitive data can be affected by malware and other attacks.	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.	**7.2** Students should incorporate clear comments in their product and document their process using text, graphics, presentations, and demonstrations.	• Students might reflect on case studies or current events in which governments or organizations experienced data leaks or data loss because of these types of attacks. • Students could give examples of potential security problems, such as denial-of-service attacks, ransomware, viruses, worms, spyware, and phishing, present threats to sensitive data.	**Network Communication & Organization** Network topology is determined, in part, by how many devices can be supported. Each device is assigned an address that uniquely identifies it on the network. The hierarchy and redundancy enable the scalability and reliability of the Internet in networks.
Practice 7	**3A-NI-08** Explain tradeoffs when selecting and implementing cybersecurity recommendations.	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	**7.2** Students should incorporate clear comments in their product and document their process using text, graphics, presentations, and demonstrations.	• 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.	**Cybersecurity** 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.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**3A-AP-20** Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.	Examples of software licenses include copyright, freeware, and many open-source licensing schemes. At previous levels, students adhered to licensing schemes. At this level, they should consider licensing implications for their work, especially when incorporating libraries and other resources.	**7.3** 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.	• Students might consider two software libraries that address a similar need, justifying their choice based on the library that has the least restrictive license.	**Program Development** Diverse teams can develop programs with a broad impact through careful review and by drawing on the strengths of members in different roles. Design decisions often involve tradeoffs, the development of complex programs aided by resources such as libraries and tools to edit and manage parts of the program. Systematic analysis is critical for identifying the effects of lingering bugs.
Practice 7	**3A-AP-23** Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.	Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program, combinations of data and procedures.	**7.2** Students should incorporate clear comments in their product and document their process using text, graphics, presentations, and demonstrations.	• Students development of complex programs aided by resources such as libraries and tools to edit and manage parts of the program	**Program Development** Diverse teams can develop programs with a broad impact through careful review and by drawing on the strengths of members in different roles. Design decisions often involve tradeoffs. Systematic analysis is critical for identifying the effects of lingering bugs.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**3A-IC-28** Explain the beneficial and harmful effects that intellectual property laws can have on innovation.	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.	**7.3** 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.	• Students could explain the 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.	**Safety Law & Ethics** 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.

#	Standards	Description	Practice Progression	Samples of student performance	Sub-Concept
Practice 7	**3A-IC-29** Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.	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 non-evident collection can raise privacy concerns, such as social media sites mining an account even when the user is not online.	**7.2** Students should incorporate clear comments in their product and document their process using text, graphics, presentations, and demonstrations.	• Students could explain 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	**Safety Law & Ethics** 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.
Practice 7	**3A-IC-30** Evaluate the social and economic implications of privacy in the context of safety, law, or ethics.	Laws govern many aspects of computing, such as privacy, data, property, information, and identity. International differences in laws and ethics have implications for computing.	**7.3** 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.	• 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	**Safety Law & Ethics** 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.

Attribution

The *Computer Science Teaching Association (CSTA) K-12 Computer Science Standards* are licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license.

The *K–12 Computer Science Framework*, led by the Association for Computing Machinery, Code.org, Computer Science Teachers Association, Cyber Innovation Center, and National

Math and Science Initiative in partnership with states and districts, is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license.

License

Except where otherwise noted, original content in the *Washington State Computer Science Standards and Practices by Grade Band Guide* by the Washington Office of Superintendent of Public Instruction is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) license.

APPENDICES

Appendix A. Computer Science State Advisory Committee

In November 2019, OSPI assembled an Advisory Committee representing a broad group of CS experts to discuss aspects of the implementation of the previously described legislative bills. They assembled in three sessions that generated rich qualitative information that was analyzed using the methodology outlined in Appendix E. The Advisory Committee members and their affiliations are listed below.

Name	Affiliation
Doug Dowell	STEM Coordinator & Grant Supervisor Central Kitsap School District
Linda Drake	Director of Career- and College-Ready Initiatives WA State Board of Education
Dr. Ellen Ebert	Science, Environmental and Sustainability Director Learning & Teaching, OSPI
Doug Ferguson	Senior Learning Designer, Interaction Design Team AVID Center
Karen Hickenbottom	Digital Learning Specialist Monroe School District
Dr. Sue Kane	Director of STEM Initiatives and Strategic Partnerships North Central Educational Service District
Greg Kilpatrick	Assistant Director of Career & Technical Education South Kitsap School District
Dr. Amy Ko	Associate Professor, Informatics Program Chair University of Washington
Juan Lozano	Instructional Specialist – CTE Highline Public Schools
Brock Maxfield	Principal Hoquiam High School
Dr. Ann McMahon	Executive Director of Research Strategy for Broad Impact Office of Research, University of Washington
Emily Rang	Director of Data Governance Center for the Improvement of Student Learning, OSPI
Dennis Small	Educational Technology Director Information Technology Services, OSPI
Dean Smith	Middle School Teacher Prosser School District
Lance Wrzesinski	Business and Marketing Program Supervisor Career & Technical Education (CTE, OSPI)

Appendix B. Frequently Asked Questions—Superintendent.

Role: Superintendent / School Board

Focus: Budget and Compliance

1. What is the practical way to get started with CS requirements?

Review the CS instruction already in place. Consider whether the high school in your district offers CS credit and whether the credit-bearing course(s) is/are open to all students. Review CS instruction at the K–8 level. Are elementary and middle schools teaching the CS skills needed for all students to access meaningful participation in a high school credit-bearing course?

2. What are other schools offering?

Each ESD has a staff person represented on the ESD CS Leadership Team. This ESD representative will be familiar with courses and instruction in many schools in your region and throughout the state.

3. Do course offerings vary according to the size of the school district?

Two important factors in determining the scope of CS instruction, especially at the high school level, are the availability of teachers with appropriate certification and the flexibility of the master schedule. In smaller school districts, both of these factors may limit options for CS instruction. Distance learning opportunities can help to overcome these limitations.

4. What's the minimum our school must do to meet state compliance?

Senate Bill 5088 requires all school districts that operate a high school to offer at least one elective course in CS that is available for all students by the 2022–23 school year. The law also allows a process approved by OSPI, which will permit students to demonstrate CS competency in lieu of completion of such a course beginning in the 2019–20 school year. Board policy must identify the credit in math or science to be awarded for all AP Computer Science courses.

Substitute House Bill 1577 further requires that all schools report the number of CS courses offered, whether the course is an advanced placement course and the number of students enrolled in CS courses. Demographic information about enrolled students is also required. Student enrollment data is reported through the school's student record system. Schools must also report the teachers that instruct CS courses and their demographic information. Teacher participation is compiled through teacher certification records and reports. This CS course, student and teacher information is required beginning in June of 2020.

For more specific detail regarding state law, please visit the *Computer Science Laws and Regulations* page of the OSPI website. URL:

https://www.k12.wa.us/student-success/resources-subject-area/computer-science/computer-science-laws-and-regulations

5. What funding is available?

Courses offered for CS credit are funded as any other basic education course, except that some (but not all) CS courses may qualify for CTE funding, in which case the course is funded by the formula of other CTE courses in the school. At this time, CS courses are not required to qualify for CTE funding, nor do they automatically do so.

6. What are the costs associated with CS courses?

Computer Science courses require hardware, software, curriculum, and teachers. These costs are typically covered by more than one single budget. Hardware is often included in the school district's overall technology plan. Software and curriculum may be locally developed or purchased from one or more third-party vendors. Instruction may be provided by a local teacher or through video or online courses. OSPI and regional ESD computer science personnel are familiar with options and services to be considered in the budgeting process.

Appendix C. Frequently Asked Questions—High School Principal.

Role: High School Principal/Academic Counselor

Focus: Courses, credits, credentials, schedules

1. What is a Computer Science course?

At the high school level, a CS course will typically be an elective course offered as a full year spanning one or more semesters/trimesters. The course will offer Computer Science credit, although dual credit options may also be available (i.e., computer science/math, business, communications, etc.)

2. What requirements must our school put in place to meet state compliance?

Senate Bill 5088 requires all school districts must provide all students with the opportunity to take at least one elective course in CS by the 2022–23 school year. Board policy must identify the credit in math or science to be awarded for all AP Computer Science courses.

Substitute House Bill 1577 further requires that all schools report the number of CS courses offered, whether the course is an advanced placement course and the number of students enrolled in CS courses. Student-related information required to be reported students' gender, grade level, demographics, eligibility for specific education programs, and a record of all courses attempted by the student. Schools must also report the teachers that instruct CS courses and their demographic information. Teacher participation is compiled through teacher certification records and reports. This CS course, student and teacher information is required beginning in June of 2020.

For more specific detail regarding state law, please visit the *Computer Science Laws and Regulations* page of the OSPI website. URL:

https://www.k12.wa.us/student-success/resources-subject-area/computer-science/computer-science-laws-and-regulations

3. What training and certification must teachers have to teach a CS course?

Refer to the Professional Educators Standards Board and OSPI certification websites for current requirements for CS teacher certification.

Continually emerging scholarship in the area of CS makes ongoing professional development for CS teachers essential. The district's Title II and Title IV funding may be possible sources for this training. Local professional development plans should include CS training for all teachers as the integration of CS with every academic discipline is an increasing priority.

4. Do CS courses qualify as CTE courses?

Some, but not all, CS courses may meet CTE requirements. At this time, CS courses are not required to meet CTE standards, nor do they automatically do so. Districts must comply with CTE rules, including submission and approval of course Frameworks and meeting teacher certification requirements.

5. What credit do students enrolled in CS courses receive?

High school level CS courses provide computer science credit as elective credit. Integrated courses that focus on both CS and math or science may offer credit in both CS and that subject. However, students must determine the subject area for which they apply the credit for the class (i.e., they cannot receive double credit for one class).

6. What are the CS curriculum options available?

Schools may offer a locally developed curriculum aligned to the state CS standards. The College Board has approved CS endorsed providers that offer multiple CS course curriculums and professional development. Video supported instruction can also be contracted through 3rd party vendors (e.g., TEALS). Online instruction may also be an option for some schools.

7. What factors should be considered to ensure that all high school students have access to a CS course?

The following students may lack meaningful access to a CS course unless their unique needs are considered:

- High Achieving students for whom other college preparatory classes may crowd out options for a CS class.
- Students that have not had an earlier interest in CS – ensure that an entry-level course does not require pre-requisites.
- Students with disabilities – plan for accommodations to increase student access and success.
- Students with "non-traditional" academic history (e.g., highly capable but underachieving students) – ensure that a low GPA does not automatically preclude enrollment.
- Students with scheduling difficulties or with significant CS experience outside the classroom.

Appendix D. Frequently Asked Questions—K–8 Principal

Role: K–8 Principal / Academic Counselor

Focus: Courses, credits, credentials, schedules

1. What is a Computer Science course?

At the K–8 level, CS will sometimes be offered as a unit or project. In this case, the district is not likely to report such instruction as a course. However, when substantial CS standards are embedded in integrated instruction that is sustained for a quarter/semester/trimester at the K–8 level, districts may choose to report this instruction as a CS course.

2. What requirements must our school put in place to meet state compliance?

Because one or more CS courses are required at the high school level, and this course(s) must be accessible for all students, students must develop the foundational skills needed beginning in elementary school in order to succeed in high school CS instruction. This is best accomplished by a systematic program of instruction throughout the K–8 learning experience.

To ensure that state reports reflect the CS instruction at the K–8 level, schools may consider defining course descriptions for CS instruction and encoding this instruction through their student record system. For more specific detail regarding state law, please visit the *Computer Science Laws and Regulations* page of the OSPI website. URL:

https://www.k12.wa.us/student-success/resources-subject-area/computer-science/computer-science-laws-and-regulations

3. How can K–8 schools fit computer science courses into tight schedules?

Schools must plan for multiple curriculum requirements and often find it challenging to teach curriculum standards in isolation. The state CS standards are appropriate for the inclusion of other content areas such as math and science. The integration of CS into related disciplines will optimize classroom scheduling and support CS instruction for all students. Instruction in the use of computer technology can be expanded to include CS foundations as well. Science classes could embed CS with a focus on the design of data collection and data analysis using digital technology; literature and communication classes could embed CS standards with an emphasis on digital communication, the impacts of social media, and the responsible use of materials created by others.

4. What training and certification must teachers have to teach a CS course?

Refer to the Professional Educators Standards Board and OSPI certification websites for current requirements for CS teacher certification.

Continually emerging scholarship in the area of CS makes ongoing professional development for CS teachers essential. The district's Title II and Title IV funding may be possible sources for this training. Local professional development plans should include CS training for all teachers as the integration of CS with every academic discipline is an increasing priority.

5. Where can our school find support in designing the K–8 CS curriculum?

Each ESD has a staff person that can help districts with CS. This ESD representative will be familiar with courses and instruction in many schools in your region and throughout the state.

6. What factors should be considered to ensure that all our students are prepared for success in CS courses in high school and beyond?

The following students may lack meaningful access to a CS course unless their unique needs are considered:

- Any student who does not express an interest in Computer Science – make instruction student-centered to allow students to connect a personal interest with CS instruction.
- Students with disabilities – plan for accommodations to increase student success.
- Students with "non-traditional" academic history (e.g., highly capable but underachieving students) – these students often need encouragement and flexibility to make a "hands on" connection to CS.

Appendix E. Methodology used in the development of this document.

A mixed-methods approach informed the development of this guidance document and supporting resources. First, an environmental scan and literature review was conducted to ground the work in research and in the learned experiences of states across the Nation. The summarized information from this task was presented in a brief and on slides to provide rich contextual background for the Advisory Committee members.

A total of three Advisory Committee meetings occurred (one in-person and two virtual) from November 2019 to April 2020. The first half-day Advisory Committee in-person meeting generated rich quantitative and qualitative data using targeted protocols with the whole group and small group divided by grade level band. In addition to discussions between the Advisory Committee members, several state leaders connected virtually with the group to shared essential learnings from the implementation of their state computer science plan. Discussions were recorded, transcribed, and uploaded into Dedoose, a software package designed for qualitative narrative analysis. Emerging themes from the analysis influenced the design of the draft guidance document. Advisory Committee members received the draft guidance document to review and provide feedback and comments.

The second Advisory Committee meeting occurred virtually in January 2020. Over two hours, each Advisory member shared their feedback and comments on the first draft document. The meeting was recorded, transcribed, and uploaded into Dedoose for qualitative narrative analysis resulting in a second draft of the guidance document.

The development team had planned to pilot the draft document on-site with three to five schools; however, the COVID-19 pandemic occurred, and the schools closed. Schools accepted an invitation for a virtual pilot, and a two-hour focus group ensued. The focus group protocol systematically walked participants through the draft guidance document allowing sufficient time for each person to respond to open-ended non-biased questions. Focus group participants included a building administrator, a records assistant, and a classroom teacher. They provided valuable information that was incorporated into the draft to increase utility to school staff and educators carrying out the work. Pilot participants represented the K–8 perspective, thus adding clarity about the requirements relative to non-high schools.

The last Advisory Committee member engagement occurred as a review and response opportunity in March 2020. Members reviewed the third draft and again provided comments that were reviewed with OSPI and other stakeholders. The final draft circulated among state leaders for review. Slight revisions were made, and the first version published in May 2020 with the acknowledgment that this is a living document that will be revised as legislation is implemented.

LEGAL NOTICE

Except where otherwise noted, this work by the *Office of Superintendent of Public Instruction* is licensed under a *Creative Commons Attribution License.*

Alternate material licenses with different levels of user permission are clearly indicated next to the specific content in the materials.

This resource may contain links to websites operated by third parties. These links are provided for your convenience only and do not constitute or imply any endorsement or monitoring by OSPI.

If this work is adapted, note the substantive changes and re-title, removing any Washington Office of Superintendent of Public Instruction logos. Provide the following attribution:

"This resource was adapted from original materials provided by the Office of Superintendent of Public Instruction. Original materials may be accessed at Computer Science Guidance page (https://www.k12.wa.us/student-success/resources-subject-area/computer-science/guidance).

OSPI provides equal access to all programs and services without discrimination based on sex, race, creed, religion, color, national origin, age, honorably discharged veteran or military status, sexual orientation including gender expression or identity, the presence of any sensory, mental, or physical disability, or the use of a trained dog guide or service animal by a person with a disability. Questions and complaints of alleged discrimination should be directed to the Equity and Civil Rights Director at 360-725-6162 or PO. Box 47200 Olympia, WA 98504-7200.

Download this material in PDF at the Computer Science Guidance page (https://www.k12.wa.us/student-success/resources-subject-area/computer-science/guidance). This material is available in alternative format upon request. Contact the Resource Center at 888-595-3276, TTY 360-664-3631. Please refer to this document number for quicker service: xx-xxxx.

ESTD
1889

All students prepared for post-secondary pathways, careers, and civic engagement.

Washington Office of Superintendent of
PUBLIC INSTRUCTION

Chris Reykdal | State Superintendent
Office of Superintendent of Public Instruction
Old Capitol Building | P.O. Box 47200
Olympia, WA 98504-7200