# California Computer Science Standards - Appendix

**Subsections**

1. [**Guide for Leadership**](#_c7y947ry8hlo)
2. [**Guide for Flexible Implementation**](#_nvmkjiqbhbav)
3. [**Guide for Instructional**](#_bvdnoxjn9w5u) **Practices Alignment**
4. [**Interdisciplinary Connections**](#_h92eu77s76x6)
5. [**Career Technical Education (CTE) Connections**](#_Career_Technical_Education)
6. [**Connections to Postsecondary Education**](#_6m3isc8qv13g)
7. [**Glossary**](#_Glossary)

1

## Guide for Leadership

### Strategies to Support Computer Science Standards Implementation

Educational leaders foster systemic change by clearly communicating a strong, compelling vision regarding the necessity of computer science education and its importance for developing college and career readiness as well as lifelong learning. For additional guidance regarding the need for computer science implementation, refer to the *Why Computer Science?* section of the introduction to the standards. Consistent messaging from leadership is vital to ensure coherence across an organization. This messaging is most effective when communicated to multiple stakeholders including administrators, teachers, paraprofessionals, parents, community members, and students. Shared vision and common language builds coherence, inspires stakeholders and promotes sustainable change. While communication of a compelling vision is vital, it must be accompanied by systems of support to build capacity of administrators, educators, community members, and students. The vision should be reflected in school board policies and resolutions, and in Local Control and Accountability Plans (LCAP) within the action and services portion of the annual update. Local education agencies are encouraged to consider adopting board resolutions supporting computer science, such as creating a computer science advisory committee. Educational leaders must also consider a sustained investment in resources to maintain vision implementation over time. Local education agencies are urged to consider their parcel tax and/or bonds to be submitted during elections and add computer science as an addendum to their district technology plan and/or district strategic plan.

The following strategies may be used to support standards implementation:

* **Communication**
  + Define a system-wide vision written in a language accessible to teachers, curriculum leaders, community members, school board members, students, parents, and families
  + Communicate the vision to stakeholders including multimedia methods (e.g., website, infographics, posters)
  + Conduct community meetings and parent nights to build awareness for computer science education and provide resources
  + Include student voice in messaging to stakeholders and encourage students to become advocates for the vision
  + Survey educators and other stakeholders to assess needs for professional learning and additional support
* **Building Capacity**
  + Ensure stakeholders can define the vision and understand their role in its implementation
  + Provide professional learning opportunities for teachers and paraeducators in alignment to vision
  + Provide professional learning for administrators and encourage them to learn alongside teachers
  + Facilitate student leadership opportunities to promote students teaching students, cross-age tutoring, and mentorships
  + Personalize job-embedded professional learning to maximize effectiveness
* **Sustainability**
  + Align computer science work to the vision, using common language to foster coherence across the system
  + Encourage educators to engage in peer observations and collaborative learning communities to share best practices
  + Foster collaboration across departments and grade levels to promote interdisciplinary connections and greater coherence within the system
  + Build partnerships between students and industry professionals and/or organizations
  + Reflect upon professional learning and implementation to guide continual improvement

### Supporting Computer Science through Professional Learning

Professional learning for educators and administrators builds collective efficacy and directly impacts student achievement. Educators who will be teaching computer science may or may not have a computer science background or certification. Therefore, care must be made to customize the professional learning experience by differentiating activities and instruction based on computer science experience or certification. Professional learning should lower educator anxiety regarding content knowledge by focusing on growth mindsets and building a safe culture of risk-taking. Professional learning experiences must always align to context of educators’ content area, instructional goals, and courses taught. The choice of programming languages or tools should follow, not dictate, the learning goals that align to classroom instruction. Ideal professional learning includes time for educators to experience learning in a pedagogically sound, learner-centered environment in which they have time to explore new concepts, followed by time to collaborate with colleagues in planning ways to apply these concepts in their own classroom. Peer observations, follow-up coaching, and time for reflection further cements the learning, builds capacity, and supports sustainable change.

Due to issues of equity within computer science education, professional learning in computer science should address instructional practices that promote inclusivity and provide universal access to all students, particularly those from underserved populations. Professional learning in computer science should also address ways to increase access depending on device availability. The standards were developed to be learned within a variety of environments with differing levels of device availability. In schools that do not offer a computing device for each student, strategies may include students working collaboratively on shared computing devices, rotating students through a computing experience via a weekly schedule within their classroom, or scheduling multiple classrooms across a campus to share a room that contains multiple computing devices.

#### Policies to Promote Computer Science

Vision and systems of support must be accompanied by updates to policy in order to achieve sustainable change. Recommended policies to promote and support the standards include:

* **Define Computer Science & Formally Adopt the Standards as Board Policy** **-** Computer science instruction should be guided by standards and align to an accurate definition of computer science.
* **Allocate Funding for Relevant, Rigorous Professional Learning & Course Development Support -** Local education agencies and schools need to dedicate funding for building capacity of educators to teach computer science, including development or purchase of course materials as well as technical infrastructure.
* **Maintain Awareness of and Support Certification Pathways -** In addition to traditional certification pathways, incentives and expedited, alternative pathways may be created to address short and long term need for computer science teachers.
* **Develop Partnerships with Higher Education Organizations -** Local education agencies and schools can benefit from creating direct pathways for preservice teachers to enter service in high need areas.
* **Create Dedicated Computer Science Positions at Local Education Agencies -** Leadership positions devoted to computer science education promote sustainable change and build capacity.
* **Require that All High Schools Offer Computer Science -** While computer science is to be available for all grades K–12, ensuring all high schools offer at least one discrete computer science course will assist in creating momentum to foster systemic change.
* **Allow Computer Science to Count as a Graduation Requirement -** Computer science courses that count toward a graduation requirement rather than as an elective result in increased participation by students.
* **Allow Computer Science to Count as an Admissions Requirement for Institutes of Higher Education -** Computer science courses that meet an admission requirement for an institute of high education will result in increased participation by students.

## Guide for Flexible Implementation

The overarching goal of the standards is guidance that fosters computer science instruction for all students. Implementation is flexible and should be based on needs of local capacity and context.

**Flexible Implementation Models for Computer Science Standards**

Opportunity for all *could* include but is not limited to one or more of the following options. Beginning at the top row, the examples move from basic exposure to broad and deep exposure at the bottom row.

| **Elementary Level** | **Middle School Level** | **High School Level** |
| --- | --- | --- |
| Integrated into the general education classroom | Integrated into math, science, and/or other subjects | Integrated into math, science, and/or other subjects |
| Integrated into an existing special classroom (e.g., media arts, computer lab, makerspace) | Integrated into an existing special classroom (e.g., media arts, computer lab, makerspace) | Introductory and/or independent course(s) |
| Independent special class (push-in or pull-out similar to models sometimes used for music, arts, etc.) | Introductory and/or independent course(s) | A menu of course options available for all students, including advanced courses (e.g., honors, AP, IB) |
| Integrated for all with additional independent enrichment course via extended hours options | Integrated for all with additional independent enrichment course elective options | Specialized courses  (e.g., game design, cybersecurity, networking, robotics) |

For a detailed description of building course pathways, view the source K–12 Computer Science Framework at <https://k12cs.org/curriculum-assessment-pathways/>.

### Integrating Computer Science

Particularly at the K–5 level, computer science education can be integrated into multiple subject classrooms to move toward an interdisciplinary approach. Computer science fits naturally into an interdisciplinary learning environment. Integration of computing experiences into multiple subject classrooms has been in existence for many years (e.g., Papert, 1980.) The standards contain interdisciplinary connection examples in grades K–8. Consider the following examples taken from the standards, each with multiple interdisciplinary connections.

* **K-2.AP.16** For example, when given images placed in a random order, students could give step-by-step commands to direct a robot, or a student playing a robot, to navigate to the images in the correct sequence. Examples of images include storyboard cards from a familiar story (CA CCSS for ELA/Literacy RL.K.2, RL.1.2, RL.2.2) and locations of the sun at different times of the day (CA NGSS: 1-ESS1-1).
* **3-5.DA.8** For example, students could create and administer electronic surveys to their classmates. Possible topics could include favorite books, family heritage, and after school activities. Students could then create digital displays of the data they have collected such as column histogram charts showing the percent of respondents in each grade who selected a particular favorite book. Finally, students could make quantitative statements supported by the data such as which books are more appealing to specific ages of students. As an extension, students could write an opinion piece stating a claim and supporting it with evidence from the data they collected. (CA CCSS for Mathematics 3.MD.3, 4.MD.4, 5.MD.2) (CA CCSS for ELA/Literacy W.3.1, W.4.1, W.5.1)
* **6-8.IC.24** For example, students could discuss the benefits and dangers of the increased accessibility of information available on the internet, and then compare this to the advantages and disadvantages of the introduction of the printing press in society. (HSS.7.8.4)

Elementary and middle schools often have greater flexibility than high school in their options for integrating computer science for all students. Options may include instructional units dedicated to computer science within technology, career exploration, or media arts classes, weekly computing classes offered as electives, or integration of computer science into other content areas. These courses could be taught by elementary education teachers with a multiple subject credential, content area teachers (e.g., mathematics, science, technology, music, art, media arts), or dedicated computer science teachers. While the implementation models vary, local education agencies need to align computer science education with the standards as covered in the curriculum alignment considerations section of the appendix.

The interdisciplinary guide section of the appendix provides general education, special classroom, and independent course educators with guidance regarding interdisciplinary connections to encourage integration and/or cross-departmental collaboration. Examples of an integrated approach could include but is not limited to: science teachers using computer science practices to analyze data in labs, math teachers creating computational artifacts with students to illustrate graphing, art teachers bringing algorithms into a discussion of filters for photographs, and science teachers having students program simulations. For additional examples, English teachers could have students create algorithms to analyze text when comparing characteristics of writing samples from various authors. Art teachers could have students write and modify software to produce digital works of art that would be impractical to create with traditional techniques. Science and health teachers could have students create and modify computational models that represent the transmission of infectious disease in order to predict changes in infected, susceptible, and recovered populations. Mathematics teachers could have students use computer algebra systems to explore problems that are realistic, yet computationally intensive and impractical to attempt using traditional techniques. Physical education teachers could have students collect and analyze data related to physical activities using computational tools and techniques. Social studies teachers could have students produce digital map sequences, computationally controlled to determine animation speeds, to illustrate changes in populations and related social issues over time. English language development and world languages teachers could engage students in collecting and analyzing data on the global impact of biliteracy.

#### Discrete Computer Science

Computer science can be taught as a discrete, independent course. For example, at the elementary level, the course can exist as a weekly push-in or pull-out program from a specialized computer science educator. In middle school, computer science can be taught as a semester long or yearlong course for a particular grade level or for all levels. In high school, computer science can be taught as a standalone course - as an introductory course, an advanced placement course, and/or a specialized course. The danger of this model is that standalone courses are often offered as electives, preventing access for those students who do not opt in to the courses. Even when using a standalone, independent model, educators are encouraged to seek out interdisciplinary connections and collaborate with colleagues in other departments to increase relevance for students. Computer science education can also be provided by enhancing pre-existing technology education credits and courses.

The core K–12 standards are designed for all students, even those in grades 9–12. While the specialty standards can aid the development of elective courses, local education agencies are encouraged to begin by offering at least one computer science course per high school. As computer science is implemented in K–8, the need for high schools to increase their rigor and expand computer science offerings will grow. High schools may build their computer science offerings by including specialty courses that can provide college entrance credit, career and technical preparation, or advanced placement courses. While these courses may be more advanced or specialized than the standards call for, the K–12 progression provides a conceptual foundation for their increasingly complex content. Local education agencies may consider whether to categorize computer science within an academic pathway, a Career Technical Education pathway, or both. It is appropriate to house computer science in both Career Technical Education and academic pathways. The early, foundational courses in a CTE program of study can be dual-coded as part of the CTE pathway as well as a math, science, or technology credit.

## Guide for Instructional Practices Alignment

### Instructional Practices Alignment Considerations

In accordance to the standards, computer science instruction and instructional practices should align to the five concepts and seven practices of computer science, grounded in solid pedagogical philosophies of student-led learning environments. A comprehensive computer science education requires inclusion of each of the concepts and practices, as defined in the Introduction. Computer science is more than coding (programming). Leaders and educators must ensure that the computer science courses and corresponding instructional resources offered in their local education agencies reflect the breadth of computer science areas via the five concepts, and also provides the opportunity for students to actively engage with the content via the seven practices.

The concepts and practices are to be integrated into instruction simultaneously in order to provide relevant, meaningful learning experiences for students. Computer science instruction should allow students to construct content knowledge through active, exploratory activities that provide opportunities to solve problems, create, collaborate, and communicate. Educators are urged to lead with learning, never with tools (e.g., programming tools, equipment, or computing languages). The five concepts should drive instructional planning, accompanied by the seven practices as a means of engaging students with the content. Tools are to be chosen not as a focus, but as a means of supporting students in mastering the concepts while engaging in practices.

Instructional practices must also take into consideration the progressive grade-bands of the framework. The standards are designed to be developmentally appropriate, building in complexity per grade-band. Careful attention to these progressions will ensure a coherent computer science education for students.

Instructional practices should reflect a myriad of socially relevant and culturally situated contexts. Students should be engaged in learning experiences that promote social connections across cultures, celebrate diversity, and allow opportunities to address relevant community issues.

### Assessment

Assessment of computer science in classrooms should be used to drive instructional planning and measure performance toward mastery of the standards. Traditional assessments that seek one solution to a problem are not recommended. The following assessment models more accurately reflect the computer science field, motivate students in meaningful learning experiences, and more accurately assess student depth of understanding.

Authentic projects allow educators to gauge not only students’ content knowledge, but their ability to apply this knowledge to real-world contexts. Throughout a long-term project, educators may use predetermined timelines as check-ins to be used as formative assessment. The assessment data collected should be used to support students according to their specific learning needs. Rubrics are recommended to ensure that students and educators have clear expectations as to the final goals of the project. This also promotes student metacognition, as learners may monitor their progress according to the rubric.

Portfolios developed over time provide students and educators the opportunity to validate the process of learning computer science, and to celebrate growth over time. As students create portfolios, they build confidence and begin to take ownership of their learning.

Educators may assess computer science concepts and skills via performance tasks. These assessments may assess multiple concepts and practices simultaneously. As performance tasks contain more than a single method and/or answer to a question, they promote student creativity, allow students to demonstrate problem solving skills, and provide educators with valuable insight into student understanding.

Assessment in computer science should contain breadth across concepts and integrate practices. The K12 Computer Science Framework, <https://k12cs.org/> contains the following examples:

For example, teachers can assess students’ ability to analyze the advantages and disadvantages of different encryption algorithms, which addresses the idea of algorithmic performance (Algorithms and Programming) as well as cybersecurity (Networks and the Internet). Even when programming is the focus, students should be assessed on not only their ability to write the program but also their ability to communicate the product’s significance and development process (Communicating About Computing), including the collaboration among members (Collaborating Around Computing). For example, students can submit planning documents used to produce the program, do a presentation on the impact that their program will have on a target audience, and write a reflection on how the team worked to put the program together.

### Universal Access

The standards are designed to be accessible to all, regardless of race, ethnicity, gender, socioeconomic status, language, religion, sexual orientation, cultural affiliation, or special needs. In addition to developing a schedule that ensures each and every student receives access to core concepts and practices, computer science learning experiences must be designed to be inclusive of all learners beginning with instructional planning. The Universal Design for Learning (UDL) framework provides a proactive, research based framework to guide educators in planning instruction that meets the varying needs of a diverse group of students. (See CAST at [http://www.cast.org](http://www.cast.org/) and National Center on Universal Design for Learning at [http://www.udlcenter.org](http://www.udlcenter.org/).) UDL principles for instruction include: Principle I) Provide multiple means of engagement to tap individual learners’ interests, challenge them appropriately, and motivate them to learn; Principle II) Provide multiple means of representation to give students various ways of acquiring, processing, and integrating information and knowledge; and Principle III) Provide multiple means of action and expression to provide students with options for navigating and demonstrating learning. The following table, developed by the *Creative Technology Research Lab* at the University of Illinois, demonstrates examples of UDL integration in computer science instruction.

| Multiple Means of **Representation** | Multiple Means of **Action and Expression** | Multiple Means of **Engagement** |
| --- | --- | --- |
| *Provide options for perception*  Model computing using physical representations as well as drawings.  Give access to modeled code while students work independently.  Provide access to video tutorials of computing tasks.  Select coding apps and websites that allow the students to adjust visual settings (such as font size & contrast) and that are compatible with screen readers. | *Provide options for physical action*  Provide teacher’s codes as templates.  Include unplugged activities that show physical relationship of abstract computing concepts.  Use assistive technology including larger/smaller mice, touch-screen devices.  Select coding apps and websites that allow coding with keyboard shortcuts in addition to dragging & dropping with a mouse. | *Provide options for recruiting interest*  Give students choices (choose project, software, topic).  Allow students to make projects relevant to culture and age and to address societal needs.  Minimize possible common “pitfalls” for both computing and content.  Allow for differences in pacing and length of work sessions.  Provide options to increase or decrease sensory stimulation (for example listening to music with headphones or using noise cancelling headphones).  Allow for differences in pacing and length of work sessions.  Allow students to address standards as part of larger projects.  Allow computer programming assignments that facilitate artistic expression. |
| *Provide options for language mathematical expressions, and symbols*  Teach and review content specific vocabulary.  Teach and review computing vocabulary (e.g., code, animations, computing, algorithm).  Post anchor charts and provide reference sheets with images of blocks or with common syntax when using text. | *Provide options for expression and communication*  Give options of unplugged activities and computing software and materials.  Give opportunities to practice computing skills and content through projects that build prior lessons.  Provide sentence starters or checklists for communicating in order to collaborate, give feedback, and explain work.  Create physical manipulatives of commands, blocks or lines of code.  Provide options that include starter code. | *Provide options for sustaining effort and persistence*  Remind students of both computing and content goals.  Provide support or extensions for students to keep engaged.  Teach and encourage peer collaboration by sharing products.  Utilize pair programming and group work with clearly defined roles.  Discuss the integral role of perseverance and problem solving in computer science. Recognize students for demonstrating perseverance and problem solving in the classroom. |
| *Provide options for comprehension*  Activate background knowledge by making computing tasks interesting and culturally relevant.  State lesson content/ computing goals.  Encourage students to ask questions as comprehension checkpoints.  Use relevant analogies and make cross-curricular connections explicit (e.g., comparing iterative product development to the writing process).  Provide graphic organizers for students to “translate” programs into pseudocode. | *Provide options for executive functions*  Guide students to set goals for long-term projects.  Record students’ progress (have planned checkpoints during lessons for understanding and progress for computing skills and content).  Provide exemplars of completed products.  Embed prompts to stop and plan, test, or debug throughout a lesson or project.  Provide graphic organizers to facilitate planning, goal-setting, and debugging.  Provide explicit instruction on skills such as asking for help, providing feedback, and using problem solving techniques.  Demonstrate debugging with think-alouds. | *Provide options for self-regulation*  Communicate clear expectations for computing tasks, collaboration, and help seeking.  Develop ways for students to self-assess and reflect on own projects and those of others.  Use assessment rubrics that evaluate both content and process.  Break-up coding activities with opportunities for reflection such as turn and talks or written questions.  Acknowledge difficulty and frustration. Model different strategies for dealing with frustration appropriately. |

Israel, M., Lash, T. A., & Jeong, G. (2017). Utilizing the Universal Design for Learning Framework in K-12 Computer Science Education. Project TACTIC: Teaching All Computational Thinking through Inclusion and Collaboration. Adapted from University of Illinois, Creative Technology Research Lab website: [https://CTRL.education.illinois.edu/TACTICal/udl](https://ctrl.education.illinois.edu/TACTICal/udl).

## Interdisciplinary Connections

Life is not divided into subject areas, and computer science is no different. As a field, computer science spans multiple disciplines.

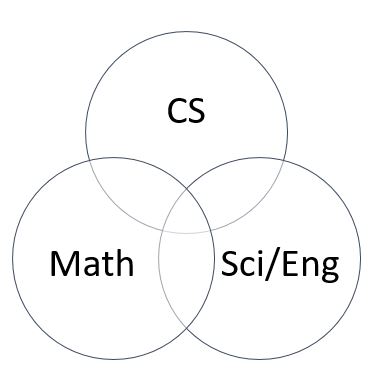
The following resources provide guidance as to interdisciplinary connections between the computer science standards and other California state board of education adopted curriculum standards. Career Technical Education (CTE) connections are listed in section V of the appendix. **The interdisciplinary connections are meant to be general suggestions as to relationships between content areas and do not constitute guidance for synonymous instruction between disciplines.** K–12 broad interdisciplinary connections followed by specific cross-disciplinary references by grade band are provided below.

### Practices

K–12 broad interdisciplinary relationships to computer science core practices:

| **Computer Science Practice** | **Related Interdisciplinary Connection(s)** |
| --- | --- |
| **1. Fostering an Inclusive Computing Culture** | **Visual and Performing Arts**  Historical and Cultural Context (Strand 3)  **Next Generation Science Standards**  Using Mathematics and Computational Thinking  (Science and Engineering Practice 5) |
| **2. Collaborating Around Computing** | **English Language Arts**  College and Career Readiness Anchor Standards for Speaking and Listening  Comprehension and Collaboration  (Standards 1, 2, 3)  **English Language Development**  Part I: Interacting in Meaningful Ways  Communicative Mode: Collaborative  (Standards 1-4)  **Next Generation Science Standards**  Using Mathematics and Computational Thinking  (Science and Engineering Practice 5) |
| **3. Recognizing and Defining Computational Problems** | **Next Generation Science Standards**  Asking Questions (for science) and defining problems (for engineering)  (Science and Engineering Practice 1)  Using Mathematics and Computational Thinking  (Science and Engineering Practice 5)  **Mathematics**  Make Sense of Problems and Persevere in Solving Them  (Standards of Mathematical Practice 1)  **Health Education Content Standards**  Decision Making  (Standard 5) |
| **4. Developing and Using Abstractions** | **Next Generation Science Standards**  Developing and Using Models  (Science and Engineering Practice 2)  **Mathematics**  Reason Abstractly and Quantitatively  (Standards of Mathematical Practice 2)  Model with Mathematics  (Standards of Mathematical Practice 4)  Look For and Make Use of Structure  (Standards of Mathematical Practice 7)  Look For and Express Regularity in Repeated Reasoning  (Standards of Mathematical Practice 8) |
| **5. Creating Computational Artifacts** | **Visual and Performing Arts**  Creative Expression (Strand 2)  **English Language Arts**  College and Career Readiness Anchor Standards for Writing  Text Types and Purposes  (Standards 1, 2, 3)  Production and Distribution of Writing  (Standards 4, 5, 6)  **Next Generation Science Standards**  Constructing Explanations (for science) and Designing Solutions (for engineering)  (Science and Engineering Practice 6) |
| **6. Testing and Refining Computational Artifacts** | **Next Generation Science Standards**  Planning and Carrying Out Investigations  (Science and Engineering Practice 3)  Developing and Using Models  (Science and Engineering Practice 2) |
| **7. Communicating About Computing** | **California History-Social Studies Framework**  Argumentative Writing  (Concept and Disciplinary Practice)  **Model School Library Standards**  Students Use Information  (Standard 3)  **Next Generation Science Standards**  Analyzing and Interpreting Data  (Science and Engineering Practice 4)  Engaging in Argument from Evidence  (Science and Engineering Practice 7)  Obtaining, Evaluating, and Communicating Information  (Science and Engineering Practice 8)  **Mathematics**  Construct Viable Arguments and Critique the Reasoning of Others  (Standards of Mathematical Practice 3)  Attend to Precision  (Standards of Mathematical Practice 6)  **English Language Arts**  College and Career Readiness Anchor Standards for Writing  Production and Distribution of Writing  (Standards 4, 5, 6)  College and Career Readiness Anchor Standards for Speaking and Listening  Comprehension and Collaboration  (Standards 1, 2, 3)  College and Career Readiness Anchor Standards for Speaking and Listening  Presentation of Knowledge and Ideas  (Standards 4, 5, 6)  College and Career Readiness Anchor Standards for Language  Vocabulary Acquisition and Use  (Standards 4, 6)  **English Language Development**  Part I: Interacting in Meaningful Ways  Communicative Mode: Productive  (Standards 9-12)  **Health Education Content Standards**  Interpersonal Communication  (Standard 4) |

The CS practices intersect with the practices described in the Next Generation Science Standards and the Common Core Standards for Mathematical Practice (see figure below).



| **Intersection CS and Math** | **Intersection CS and Science/Engineering** | **Intersection CS, Math, and Science/Engineering** |
| --- | --- | --- |
| **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 | **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 | **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 |

Adapted from: K-12 CS Framework, p. 72.

<https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf>

Communicating about Computing is a key practice that is reinforced throughout the CS standards by asking students to describe, explain, and justify program design decisions, computing phenomena, and the impacts of computing on society. These CS standards support numerous ELA speaking, listening, and writing standards as well as ELD standards, including synthesizing information from multiple sources, writing discipline-specific arguments, developing facility with technical vocabulary, and understanding diverse perspectives.

### Interdisciplinary Connections to Standards by Grade Band

Note: Cross-disciplinary references from the interdisciplinary examples listed in the standards document may or may not be included on this table, as the connections listed below are meant to be general connections rather than aligned to specific examples.

### Grades K–2

Key: Null symbol (Ø) = not found

| **Computer Science Standard** | CA CCSS for ELA/Literacy | CA English Language Development | History Social Studies | CA CCSS for Mathematics | Visual and Performing Arts | CA Next Generation Science | CA Model School Library Standards | Physical Ed | CA Health Ed |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| K-2.CS.1 | [W.K.6](http://www.corestandards.org/ELA-Literacy/W/K/6/), W.1.6, W.2.6, SL.K.5, SL.1.5, SL.2.5 | P.I.K.2, P.I.K.9, P.I.K.10, P.I.K.12, P.I.1.2, P.I.1.9, P.I.1.10, P.I.1.12, P.I.2.2, P.I.2.9, P.I.2.10, P.I.2.12 | Ø | Ø | Ø | Ø | K.1.3.G, K.1.4.A, K.3.3, K.4.1, 1.1.3.A, 1.1.4, 1.3.3, 1.4.1, 2.1.3.B, 2.1.3.D, 2.1.4, 2.4.3.B | Ø | Ø |
| K-2.CS.2 | SL.K.4, SL.K.5, SL.K.6, SL.1.4, SL.1.5, SL.1.6, SL.2.4, SL.2.5, SL.2.6 | P.I.K.1, P.I.K.3, P.I.K.9, P.I.K.11, PI.K.12, P.II.K.1, P.II.K.2, P.II.K.3, P.II.K.4, P.II.K.5, P.I.1.3, P.I.1.9, P.I.1.11, P.I.1.12, P.II.1.1, P.II.1.2, P.II.1.3, P.II.1.4, P.II.1.5, P.II.1.6, P.II.1.7, P.I.2.1, P.I.2.3, P.I.2.4, P.I.2.9, P.I.2.11, P.I.2.12, P.II.2.1; P.II.2.2, P.II.2.3, P.II.2.4, P.II.2.5, P.II.2.6, P.II.2.7 | Ø | Ø | Ø | K-2-ETS1-2 | 1.1.3.C, 2.3.3.B, 2.3.3.D | Ø | Ø |
| K-2.CS.3 | SL.K.4, SL.K.5, SL.K.6, SL.1.4, SL.1.5, SL.1.6, SL.2.4, SL.2.5, SL.2.6 | P.I.K.1, P.I.K.3, P.I.K.9, P.I.K.11, PI.K.12, P.II.K.1, P.II.K.2, P.II.K.3, P.II.K.4, P.II.K.5, P.I.1.3, P.I.1.9, P.I.1.11, P.I.1.12, P.II.1.1, P.II.1.2, P.II.1.3, P.II.1.4, P.II.1.5, P.II.1.6, P.II.1.7, P.I.2.1, P.I.2.3, P.I.2.4, P.I.2.9, P.I.2.11, P.I.2.12, P.II.2.1, P.II.2.2, P.II.2.3, P.II.2.4, P.II.2.5, P.II.2.6, P.II.2.7 | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| K-2.NI.4 | SL.K.4, SL.K.5, SL.K.6, SL.1.4, SL.1.5, SL.1.6, SL.2.4, SL.2.5, SL.2.6 | P.I.K.1, P.I.K.3, P.I.K.9, P.I.K.11, PI.K.12, P.II.K.1, P.II.K.2, P.II.K.3, P.II.K.4, P.II.K.5, P.I.1.3, P.I.1.9, P.I.1.11, P.I.1.12, P.II.1.1, P.II.1.2, P.II.1.3, P.II.1.4, P.II.1.5, P.II.1.6, P.II.1.7, P.I.2.1, P.I.2.3, P.I.2.4, P.I.2.9, P.I.2.11, P.I.2.12, P.II.2.1, P.II.2.2, P.II.2.3, P.II.2.4, P.II.2.5, P.II.2.6, P.II.2.7 | Ø | Ø | Ø | 1-PS-4-4, K-2-ETS1-2 | 1.3.1.B | Ø | Ø |
| K-2.NI.5 | SL.K.4, SL.K.5, SL.K.6, SL.1.4, SL.1.5, SL.1.6, SL.2.4, SL.2.5, SL.2.6 | P.I.K.1, P.I.K.3, P.I.K.9, P.I.K.11, PI.K.12, P.II.K.1, P.II.K.2, P.II.K.3, P.II.K.4, P.II.K.5, P.I.1.3, P.I.1.9, P.I.1.11, P.I.1.12, P.II.1.1, P.II.1.2, P.II.1.3, P.II.1.4, P.II.1.5, P.II.1.6, P.II.1.7, P.I.2.1, P.I.2.3, P.I.2.4, P.I.2.9, P.I.2.11, P.I.2.12, P.II.2.1, P.II.2.2, P.II.2.3, P.II.2.4, P.II.2.5, P.II.2.6, P.II.2.7 | Ø | Ø | Ø | Ø | K.3.1.A, 2.3.1.E | Ø | Ø |
| K-2.NI.6 | Ø | Ø | Ø | Ø | Music.K.1.1 | 1-PS-4-4 | Ø | Ø | Ø |
| K-2.DA.7 | W.K.5, W.K.6, W.1.5, W.1.6, W.2.5, W.2.6 | PI.K.2, PI.K.10, PI.K.12, PII.K.1, PII.K.2, PII.K.3, PII.K.4, PII.K.5, PII.K.6, PI.1.2, PI.1.10, PI.1.12, PII.1.1, PII.1.2, PII.1.3, PII.1.4, PII.1.5, PII.1.6, PII.1.7, PI.2.2, PI.2.4, PI.2.10, PI.2.12, PII.2.1, PII.2.3, PII.2.4, PII.2.5, PII.2.6, PII.2.7 | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| K-2.DA.8 | Ø | Ø | Ø | K.MD.3, 1.MD.4,  2.MD.4  2.MD.9  2.MD.10 | Ø | K-2-ETS1-3, K-PS2-2, K-LS1-1, K-ESS2-1, 1-ESS1-1, 1-ESS1-2, 2-PS1-2 | Ø | Ø | Ø |
| K-2.DA.9 | Ø | PI.K.9, PI.K.12b, PII.K.2, PII.K.5, PII.K.6, PI.1.9, PI.1.12b, PII.1.2, PII.1.5, PII.1.6, PI.2.9, PI.2.12b, PII.2.2, PII.2.5, PII.2.6 | Ø | K.MD.3,  K.G.4, 1.MD.4, 2.MD.10 | Ø | 2-PS1-1, 2-PS1-2, K-PS3-1, K-ESS2-1, K-ESS3-2, 1-ESS1-1, 1-ESS1-2 | Ø | Ø | Ø |
| K-2.AP.10 | W.K.3, W.1.3, W.2.3 | PI.K.10, PII.K.1, PII.K.2, PII.K.6, PI.1.10, PII.1.1, PII.1.2, PII.1.6, PII.1.7, PI.2.10, PII.2.1, PII.2.2, II.2.6, PII.2.7 | K.5 | Ø | Dance 1.2.3, Dance 1.2.5, Dance 2.2.3 | Ø | Ø | 2.1.18, 2.1.19 | Ø |
| K-2.AP.11 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| K-2.AP.12 | Ø | Ø | Ø | Ø | Dance 1.2.3, Dance 1.2.5, Dance 2.2.3 | Ø | Ø | 2.1.18, 2.1.19 | Ø |
| K-2.AP.13 | Ø | Ø | Ø | K.G.5, 1.G.2, 2.NBT.9 | Ø | 2-PS1-3 | Ø | Ø | 1.6.1.P |
| K-2.AP.14 | W.K.3, W.1.3, W.2.3 | PI.K.10, PII.K.1, PII.K.2, PII.K.6, PI.1.10, PII.1.1, PII.1.2, PII.1.6, PII.1.7, PI.2.10, PII.2.1, PII.2.2, PII.2.6, PII.2.7 | Ø | Ø | Ø | Ø | Ø | Ø | K.7.2.N, K.6.1.M, 1.6.1.P, 2.7.2.N |
| K-2.AP.15 | Ø | PI.K.9, PI.1.9, PI.2.9, PI.K.10, PI.1.10, PI.2.10 | Ø | Ø | Ø | Ø | 2.3.1.B | Ø | Ø |
| K-2.AP.16 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| K-2.AP.17 | SL.K.4, SL.K.5, SL.K.6, SL.1.4, SL.1.5, SL.1.6, SL.2.4, SL.2.5, SL.2.6 | P.I.K.1, P.I.K.3, P.I.K.9, P.I.K.11, PI.K.12, P.II.K.1, P.II.K.2, P.II.K.3, P.II.K.4, P.II.K.5, P.I.1.3, P.I.1.9, P.I.1.11, P.I.1.12, P.II.1.1, P.II.1.2, P.II.1.3, P.II.1.4, P.II.1.5, P.II.1.6, P.II.1.7, P.I.2.1, P.I.2.3, P.I.2.4, P.I.2.9, P.I.2.11, P.I.2.12, P.II.2.1, P.II.2.2, P.II.2.3, P.II.2.4, P.II.2.5, P.II.2.6, P.II.2.7 | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| K-2.IC.18 | SL.K.4, SL.K.5, SL.K.6, SL.1.4, SL.1.5, SL.1.6, SL.2.4, SL.2.5, SL.2.6 | P.I.K.1, P.I.K.3, P.I.K.9, P.I.K.11, PI.K.12, P.II.K.1, P.II.K.2, P.II.K.3, P.II.K.4, P.II.K.5, P.I.1.3, P.I.1.9, P.I.1.11, P.I.1.12, P.II.1.1, P.II.1.2, P.II.1.3, P.II.1.4, P.II.1.5, P.II.1.6, P.II.1.7, P.I.2.1, P.I.2.3, P.I.2.4, P.I.2.9, P.I.2.11, P.I.2.12, P.II.2.1, P.II.2.2, P.II.2.3, P.II.2.4, P.II.2.5, P.II.2.6, P.II.2.7 | K.6.3, 1.4, 2.1, 2.4 | Ø | Ø | Ø | Ø | Ø | Ø |
| K-2.IC.19 | W.K.5, W.K.5, W.1.5, W.1.6, W.2.5, W.2.6 | PI.K.2, PI.K.10, PI.K.12, PII.K.1, PII.K.2, PII.K.3, PII.K.4, PII.K.5, PII.K.6, PI.1.2, PI.1.10, PI.1.12, PII.1.1, PII.1.2, PII.1.3, PII.1.4, II.1.5, PII.1.6, PII.1.7, PI.2.2, PI.2.4, PI.2.10, PI.2.12, PII.2.1, PII.2.3, PII.2.4, PII.2.5, PII.2.6, PII.2.7 | K.1.1, 1.1.2 | Ø | Ø | Ø | 2.3.1.A | 2.5.2, 2.5.3, 2.5.4 | K.1.1.S, K.1.6.S, K.1.7.S, K.4.2.M, K.7.2.M, K.8.1.M, 1.1.6.S, 1.8.1.S, 2.1.10.M |
| K-2.IC.20 | SL.K.4, SL.K.5, SL.K.6, SL.1.4, SL.1.5, SL.1.6, SL.2.4, SL.2.5, SL.2.6 | P.I.K.1, P.I.K.3, P.I.K.9, P.I.K.11, PI.K.12, P.II.K.1, P.II.K.2, P.II.K.3, P.II.K.4, P.II.K.5, P.I.1.3, P.I.1.9, P.I.1.11, P.I.1.12, P.II.1.1, P.II.1.2, P.II.1.3, P.II.1.4, P.II.1.5, P.II.1.6, P.II.1.7, P.I.2.1, P.I.2.3, P.I.2.4, P.I.2.9, P.I.2.11, P.I.2.12, P.II.2.1, P.II.2.2, P.II.2.3, P.II.2.4, P.II.2.5, P.II.2.6, P.II.2.7 | Ø | Ø | Ø | Ø | 1.1.3.F, 2.3.1.E | Ø | Ø |

Note: Cross-disciplinary references from the interdisciplinary examples listed in the standards document may or may not be included on this table, as the connections listed above are meant to be general connections rather than aligned to specific examples.

### Grades 3–5

Key: Null symbol (Ø) = not found

| **Computer Science Standard** | CA CCSS for ELA/Literacy | CA English Language Development | History Social Studies | CA CCSS for Mathematics | Visual and Performing Arts | CA Next Generation Science | CA Model School Library Standards | Physical Education | Health Education |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3-5.CS.1 | SL.3.4, SL.4.4, SL.5.4 | PI.3.9, PI.3.12, PI.4.9, PI.4.12, PI.5.9, PI.5.12 | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 3-5.CS.2 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 3-5.CS.3 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 3-5.NI.4 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 3-5.NI.5 | SL.3.4, SL.4.4, SL.5.4 | PI.3.9, PI.3.12, PI.4.9, PI.4.12, PI.5.9, PI.5.12 | Ø | Ø | Ø | Ø | 3.3.1.B, 4.3.1.B, 5.3.1.C, 5.3.1.E, 5.3.1.F | Ø | 3.1.4.M, 4.1.1.S |
| 3-5.NI.6 | Ø | Ø | Ø | Ø | Ø | 4-PS4-3 | Ø | Ø | Ø |
| 3-5.DA.7 | W.3.6, W.4.6, W.5.6, SL.3.4, SL.4.4, SL.5.4 | PI.3.9, PI.3.12, PI.4.9, PI.4.12, PI.5.9, PI.5.12 | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 3-5.DA.8 | Ø | Ø | Ø | 3.MD.3, 4.MD.4, 5.MD.2 | Ø | 3-PS2-2, 3-LS3-1, 3-LS4-1, 3-ESS2-1, 4-ESS2-1, 4-ESS2-2, 5-PS1-2, 5-ESS1-2, 5-ESS2-2 | Ø | Ø | Ø |
| 3-5.DA.9 | Ø | Ø | Ø | 3.MD.3, 4.MD.4, 5.MD.2 | Ø | 3-PS2-2, 3-LS3-1, 3-LS4-1, 3-ESS2-1, 4-ESS2-1, 4-ESS2-2, 5-PS1-2, 5-ESS1-2, 5-ESS2-2 | Ø | Ø | Ø |
| 3-5.AP.10 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 3-5.AP.11 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 3-5.AP.12 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 3-5.AP.13 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 3-5.AP.14 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 3-5.AP.15 | Ø | Ø | Ø | Ø | Ø | 3-5-ETS1-1, 3-5-ETS1-2 | Ø | Ø | Ø |
| 3-5.AP.16 | Ø | Ø | Ø | Ø | Ø | Ø | 3.1.4.A, 5.3.1.A, 5.3.1.B, 5.3.1.C | Ø | Ø |
| 3-5.AP.17 | Ø | Ø | Ø | Ø | Ø | 3-5-ETS1-3 | Ø | Ø | Ø |
| 3-5.AP.18 | Ø | Ø | Ø | Ø | Theatre 5.2.3, Theatre 5.5.2 | Ø | Ø | 3.5.6, 4.5.6, 5.5.5 | Ø |
| 3-5.AP.19 | SL.3.4, SL.4.4, SL.5.4 | PI.3.9, PI.3.12, PI.4.9, PI.4.12, PI.5.9, PI.5.12 | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 3-5.IC.20 | SL.3.1, SL.4.1, SL.5.1 | PI.3.1, PI.3.5, PI.4.1, PI.4.5, PI.5.1, PI.5.5 | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 3-5.IC.21 | W.3.7, W.4.7, W.5.7 | Ø | Ø | Ø | Ø | 3-5-ETS1-1, 3-5-ETS1-2, 3-5-ETS1-3 | Ø | Ø | Ø |
| 3-5.IC.22 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 3-5.IC.23 | Ø | Ø | Ø | Ø | Ø | Ø | 3.1.4.A, 5.3.1.B, 5.3.1.E | Ø | Ø |

Note: Cross-disciplinary references from the interdisciplinary examples listed in the standards document may or may not be included on this table, as the connections listed above are meant to be general connections rather than aligned to specific examples.

### Grades 6–8

Key: Null symbol (Ø) = not found

| **Computer Science Standard** | CA CCSS for ELA/Literacy | CA English Language Development | History Social Studies | CA CCSS for Mathematics | Visual and Performing Arts | CA Next Generation Science | CA Model School Library Standards | Physical Education | Health Education |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6-8.CS.1 | Ø | Ø | Ø | 7.SP.8 | Ø | MS-ETS1-1 | Ø | Ø | 6.7.3.M |
| 6-8.CS.2 | Ø | Ø | Ø | 8.F.5 | Ø | MS-ETS1-1, MS-ETS1-3 | Ø | Ø | Ø |
| 6-8.CS.3 | Ø | Ø | Ø | Ø | Ø | MS-ETS1-4 | Ø | Ø | Ø |
| 6-8.NI.4 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 6-8.NI.5 | Ø | Ø | Ø | Ø | Ø | Ø | 6.3.1.A, 6.3.1.B, 6.3.1.D | Ø | Ø |
| 6-8.NI.6 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 6-8.DA.7 | Ø | Ø | Ø | 6.RP.3, 6.EE.9, 6.NS.1, 6.SP.4, 8.EE.5, 8.F.2, 8.F.4, 8.F.5, 8.SP.1 | Ø | MS-ETS1-3 | Ø | Ø | Ø |
| 6-8.DA.8 | Ø | Ø | Ø | 7.SP.3, 8.SP.1, 8.SP.4, 7.SP.4, 8.F.4, 8.F.5, 8.SP.1, 6.SP.5, 7.RP.2, 7.SP.2, 7.SP.6 | Ø | MS-PS1-2, MS-LS4-3, MS-ESS1-3, MS-ESS2-3, MS-ESS2-5, MS-ESS3-2, MS-ETS1-3 | Ø | Ø | Ø |
| 6-8.DA.9 | Ø | Ø | Ø | 7EE.4, 7.SP.7, 8.SP.1, 8.SP.3, 8.SP.4, 8.G.9 | Ø | MS-ETS1-4, MS-PS1-2, MS-ETS1-1, MS-ETS1-2, MS-ETS1-3 | Ø | Ø | Ø |
| 6-8.AP.10 | Ø | Ø | 7.6.7, 8.2.6, 8.5.3, 8.6.5 | Ø | Ø | MS-ETS1-4 | Ø | Ø | Ø |
| 6-8.AP.11 | Ø | Ø | Ø | 6.EE.2, 6.EE.6, 7.EE.4, 8.EE.8, 8.F.1 | Ø | Ø | Ø | Ø | Ø |
| 6-8.AP.12 | Ø | Ø | Ø | 7.SP.8, 8.F.4, 8.SP.4, | Ø | MS-ETS1-4, MS-PS1-6 | Ø | Ø | Ø |
| 6-8.AP.13 | Ø | Ø | 7.6.4, 7.8.5, 7.9, 8.2.7, 8.3.6, | 6.EE.6, 7.G.2, 8.EE.8, | Ø | MS-ETS1-1, MS-ETS1-2 | Ø | Ø | Ø |
| 6-8.AP.14 | Ø | Ø | Ø | 7.EE.4 | Ø | Ø | Ø | Ø | Ø |
| 6-8.AP.15 | SL.6.1, SL.7.1, SL.8.1 | PI.6.1, PI.6.5, PI.7.1, PI.7.5, PI.8.1, PI.8.5 | Ø | Ø | Ø | MS-ETS1-1, MS-ETS1-4 | Ø | Ø | Ø |
| 6-8.AP.16 | RI.6.7, W.6.8, W.7.8, W.8.8 | PI.6.10, PI.7.10, PI.8.10 | Ø | Ø | Ø | Ø | 6.1.4.B, 7-8.3.1.D | Ø | Ø |
| 6-8.AP.17 | Ø | Ø | Ø | 6.EE.5 | Ø | MS-ETS1-2, MS-ETS1-3, MS-ETS1-4 | Ø | Ø | Ø |
| 6-8.AP.18 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | 8.5.5 | Ø |
| 6-8.AP.19 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 6-8.IC.20 | Ø | Ø | 7.3.5, 8.6.1 | Ø | Ø | Ø | 6.1.3.G | Ø | 6.3.4.A, 7-8.2.1.N, 7-8.2.4.N, 7-8.2.2.G, 7-8.4.7.S, 7-8.2.4.P, 7-8.3.3.P |
| 6-8.IC.21 | RI.6.7, RI.7.7, RI.8.7, SL.6.1, SL.7.1, SL.8.1 | PI.6.1, PI.6.5, PI.7.1, PI.7.5, PI.8.1, PI.8.5 | Ø | Ø | Ø | Ø | Ø | Ø | 6.1.5.M, 6.7.3.M, 7-8.1.5.M |
| 6-8.IC.22 | Ø | Ø | Ø | Ø | Ø | Ø | 6.4.2.C | Ø | Ø |
| 6-8.IC.23 | Ø | Ø | Ø | Ø | Ø | Ø | 6.3.1.H, 7-8.3.1.A, 7-8.3.1.B, 7-8.3.1.E, 7-8.4.2.A, 7-8.4.2.B | Ø | Ø |
| 6-8.IC.24 | Ø | Ø | Ø | Ø | Ø | Ø | 7-8.3.1.A, 7-8.3.1.B, 7-8.3.1.E, 7-8.4.2.A, 7-8.4.2.B | Ø | 6.1.7.M |

Note: Cross-disciplinary references from the interdisciplinary examples listed in the standards document may or may not be included on this table, as the connections listed above are meant to be general connections rather than aligned to specific examples.

### Grades 9–12

There is a relationship between the computer science and content standards in other subject areas, particularly in the Earth and Human Activity (ESS3) and Engineering Design (ETS1) disciplinary core ideas (DCIs) of the Next Generation Science Standards (NGSS). Although the NGSS standards are more specific to scientific domains, both sets of standards ask students to generate, evaluate, and refine computational models or simulations, and decompose problems into smaller components to facilitate designing solutions.

High school mathematics content will support student learning in computer science. High school standards in Algebra and Functions will support students in understanding, creating, and refining computational models and applying encryption techniques in computer science. High school standards in Statistics and Probability, particularly standards about interpreting data and making inferences will provide conceptual grounding for computer science standards in the Data and Analysis concept area.

The table below provides a detailed look the connections between standards in core academic disciplines and computer science.

Key: Null symbol (Ø) = not found

| **CA CS Standard Identifier** | **CA NGSS** | **CA CCSS for Mathematics** | **CA CCSS for ELA/Literacy** | **CA English Language Development** | **CA Model School Library** | **CA History Social Studies** | **CA Health Education** | **CA VAPA** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9-12.CS.1 | Ø | Ø | W.9-10.2 and W.11-12.2, L.9-10.6 and L.11-12.6, SL.9-10.4 and SL.11-12.4, WHST.9-10.2 and WHST.11-12.2 | PI.9-12.9, PI.9-12.10, PI.9-12.12 | Ø | Ø | Ø | Ø |
| 9-12.CS.2 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.CS.3 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.NI.4 | Ø | Ø | SL.9-10.1 and SL.11-12.1, SL.9-10.4 and SL.11-12.4 | PI.9-12.1, PI.9-12.5, PI.9-12.9, PI.9-12.10, PI.9-12.12 | Ø | Ø | Ø | Ø |
| 9-12.NI.5 | Ø | Ø | W.9-10.2 and W.11-12.2, L.9-10.6 and L.11-12.6, SL.9-10.4 and SL.11-12.4 | PI.9-12.9, PI.9-12.10, PI.9-12.12 | Ø | Ø | Ø | Ø |
| 9-12.NI.6 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.NI.7 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.DA.8 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.DA.9 | Ø | Ø | W.9-10.2 and W.11-12.2, L.9-10.6 and L.11-12.6, WHST.11-12.2, SL.9-10.4 and SL.11-12.4 | PI.9-12.9, PI.9-12.10, PI.9-12.12 | Ø | Ø | Ø | Ø |
| 9-12.DA.10 | Ø | S-ID.1, S-ID.5, S-ID.6 | RST.9-10.7, RST.11-12.7 | Ø | 9-12 3.3 | Ø | Ø | Ø |
| 9-12.DA.11 | HS-PS3-1, HS-ESS3-3, HS-ESS3-6, HS-ETS1-4 | S-IC.2 | Ø | Ø | 9-12 3.3 | Ø | Ø | Ø |
| 9-12.AP.12 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.AP.13 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.AP.14 | Ø | Ø | W.9-10.1 and W.11-12.1, L.9-10.6 and L.11-12.6, WHST.9-10.1 and WHST.11-12.1 | PI.9-12.11, PI.9-12.12 | Ø | Ø | Ø | Ø |
| 9-12.AP.15 | HS-ESS3-4 | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.AP.16 | HS-ETS1-2 | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.AP.17 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.AP.18 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.AP.19 | Ø | Ø | W.9-10.2 and W.11-12.2, L.9-10.6 and L.11-12.6, WHST.11-12.2, SL.9-10.4 and SL.11-12.4 | PI.9-12.9, PI.9-12.10, PI.9-12.12 | 9-12 3.1 | 12.2.2 | Ø | Ø |
| 9-12.AP.20 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.AP.21 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.AP.22 | Ø | Ø | W.9-10.2, L.9-10.6 and L.11-12.6, WHST.11-12.2, SL.9-10.4 and SL.11-12.4 | PI.9-12.9, PI.9-12.10, PI.9-12.12 | Ø | Ø | Ø | Ø |
| 9-12.IC.23 | HS-ESS3-4 | Ø | RST.11-12.7, RST.11-12.9 | Ø | 9-12 2.1, 9-12 2.2, 9-12 2.3, 9-12 3.2 | Ø | Ø | Visual Arts 9-12.5.1 |
| 9-12.IC.24 | Ø | Ø | W.9-10.7 and W.11-12.7, W.9-10.8 and W.11-12.8, W.9-10.9 and W.11-12.9, L.9-10.6 and L.11-12.6 | PI.9-12.9, PI.9-12.10, PI.9-12.12 | Ø | Ø | Ø | Ø |
| 9-12.IC.25 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.IC.26 | HS-ESS3-4 | Ø | RST.11-12.7, RST.11-12.9 | Ø | 9-12 2.1, 9-12 2.2, 9-12 2.3, 9-12 3.2 | Ø | Ø | Visual Arts 9-12.5.1 |
| 9-12.IC.27 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12.IC.28 | Ø | Ø | W.9-10.2 and W.11-12.2, L.9-10.6 and L.11-12.6, WHST.11-12.2, SL.9-10.4 and SL.11-12.4 | PI.9-12.9, PI.9-12.10, PI.9-12.12 | 9-12 3.1 | 12.2.2 | Ø | Ø |
| 9-12.IC.29 | Ø | Ø | W.9-10.2 and W.11-12.2, L.9-10.6 and L.11-12.6, WHST.11-12.2, SL.9-10.4 and SL.11-12.4 | PI.9-12.9, PI.9-12.10, PI.9-12.12 | Ø | Ø | Ø | Ø |
| 9-12.IC.30 | Ø | Ø | RST.11-12.7, RST.11-12.9, WHST.9-10.1 and WHST.11-12.1 | Ø | 9-12 2.1, 9-12 2.2, 9-12 2.3, 9-12 3.1 | Ø | Ø | Ø |
| 9-12S.CS.1 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.CS.2 | Ø | Ø | W.9-10.2 and W.11-12.2, L.9-10.6 and L.11-12.6, WHST.11-12.2, SL.9-10.4 and SL.11-12.4 | PI.9-12.9, PI.9-12.10, PI.9-12.12 | Ø | Ø | Ø | Ø |
| 9-12S.NI.3 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.NI.4 | Ø | Ø | W.9-10.2 and W.11-12.2, L.9-10.6 and L.11-12.6, WHST.11-12.2, SL.9-10.4 and SL.11-12.4 | PI.9-12.9, PI.9-12.10, PI.9-12.12 | Ø | Ø | Ø | Ø |
| 9-12S.NI.5 | Ø | Ø | Ø | Ø | 9-12 3.1 | Ø | Ø | Ø |
| 9-12S.NI.6 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.DA.7 | Ø | S-IC.3 | Ø | Ø | 9-12 3.3 | Ø | Ø | Ø |
| 9-12S.DA.8 | HS-ESS3-5 | S-IC.1, S-IC.4, S-IC.5 | Ø | Ø | 9-12 3.3 | Ø | Ø | Ø |
| 9-12S.DA.9 | HS-LS2-1, HS-ESS1-4 | S-IC.2, S-IC.5 | RST.11-12.7, RST.11-12.9, WHST.9-10.1 and WHST.11-12.1 | Ø | 9-12 2.1, 9-12 2.2, 9-12 2.3, 9-12 3.2 | Ø | Ø | Ø |
| 9-12S.AP.10 | Ø | Ø | W.9-10.2 and W.11-12.2, L.9-10.6 and L.11-12.6 | PI.9-12.10 | Ø | Ø | Ø | Ø |
| 9-12S.AP.11 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.12 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.13 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.14 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.15 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.16 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.17 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.18 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.19 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.20 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.21 | Ø | Ø | Ø | Ø | 9-12 3.1 | Ø | Ø | Ø |
| 9-12S.AP.22 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.23 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.24 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.25 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.AP.26 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø |
| 9-12S.IC.27 | HS-ESS3-3, HS-ESS3-4, HS-ESS3-6, HS-ETS1-4 | Ø | RST.11-12.7, RST.11-12.9, WHST.9-10.1 and WHST.11-12.1 | Ø | 9-12 2.1, 9-12 2.2, 9-12 2.3, 9-12 3.2 | Ø | Ø | Ø |
| 9-12S.IC.28 | Ø | Ø | RST.11-12.7, RST.11-12.9 | Ø | 9-12 2.1, 9-12 2.2, 9-12 2.3, 9-12 3.2 | Ø | Ø | Ø |
| 9-12S.IC.29 | Ø | Ø | RST.11-12.7, RST.11-12.9, WHST.9-10.1 and WHST.11-12.1 | Ø | 9-12 2.1, 9-12 2.2, 9-12 2.3, 9-12 3.2 | Ø | Ø | Ø |
| 9-12S.IC.30 | Ø | Ø | RST.11-12.8, WHST.9-10.1 and WHST.11-12.1, SL.9-10.1 and SL.11-12.1 | PI.9-12.11 | 9-12 2.1, 9-12 2.1, 9-12 2.2, 9-12 2.3, 9-12 3.1, 9-12 3.2 | 12.2.2 | Ø | Ø |

Note: Cross-disciplinary references from the interdisciplinary examples listed in the standards document may or may not be included on this table, as the connections listed above are meant to be general connections rather than aligned to specific examples.

## 

## Career Technical Education (CTE) Connections

California’s Career Technical Education (CTE) standards are separated into 15 industry sectors, with separate career pathways defined within each sector, and were adopted by the SBE on January 16, 2013. The CS practices are synergistic with the CTE standards for career ready practices and can be adopted by both academic and CTE teachers. Both the CTE and CS standards are founded in an academic discipline and provide career preparation. Both sets of practices emphasize clear communication, collaboration, understanding of diverse viewpoints, and problem-solving skills. There are two notable differences between the CS and the CTE standards. First, the CS standards identify foundational knowledge for all students - starting at the Kindergarten level - in preparation for college and career readiness. Second, while the CTE standards emphasize specific technologies and industry protocols, the CS standards strive to be technology agnostic.

There are several CTE sectors and pathways that overlap with the CS standards:

* The Information and Communication Technologies (ICT) sector
* The Game Design and Integration pathway within the Arts, Media, and Entertainment (AME) sector
* The Telecommunications pathway within the Energy, Environment, and Utilities (EEU) sector
* The Public Safety pathway within the Public Services (PS) sector

The tables below provide a detailed look at the connections between the CTE and CS standards and practices.

### CTE Standards for Career Ready Practice

Key: Null symbol (Ø) = not found

| **CTE Standards for Career Ready Practice** | **Computer Science Practices** |
| --- | --- |
| 1. Apply appropriate technical skills and academic knowledge | Alignment listed within individual sectors |
| 2. Communicate clearly, effectively, and with reason | **Practice 7.2** Describe, justify, and document computational processes and solutions using appropriate terminology consistent with the intended audience and purpose. |
| 3. Develop an education and career plan aligned with personal goals | Ø |
| 4. Apply technology to enhance productivity | **Practice 2.4** Evaluate and select technological tools that can be used to collaborate on a project.  **Practice 3.1** Identify complex, interdisciplinary, real-world problems that can be solved computationally.  **Practice 3.3** Evaluate whether it is appropriate and feasible to solve a problem computationally. |
| 5. Utilize critical thinking to make sense of problems and persevere in solving them | **Practice 3.2** Decompose complex real-world problems into manageable subproblems that could integrate existing solutions or procedures.  **Practice 4.1** Extract common features from a set of interrelated processes or complex phenomena.  **Practice 4.3** Create modules and develop points of interaction that can apply to multiple situations and reduce complexity.  **Practice 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. |
| 6. Practice personal health and understand financial literacy | Ø |
| 7. Act as a responsible citizen in the workplace and the community | **Practice 7.3** Articulate ideas responsibly by observing intellectual property rights and giving appropriate attribution. |
| 8. Model integrity, ethical leadership, and effective management | **Practice 2.2** Create team norms, expectations, and equitable workloads.  **Practice 2.3** Solicit and incorporate feedback from, and provide constructive feedback to, team members and other stakeholders. |
| 9. Work productively in teams while integrating cultural and global competence | **Practice 2.1** Cultivate working relationships with individuals possessing diverse perspectives, skills, and personalities. |
| 10. Demonstrate creativity and innovation | **Practice 5.2** Create a computational artifact for practical intent, personal expression, or to address a societal issue. |
| 11. Employ valid and reliable research strategies | **Practice 4.4** Model phenomena and processes and simulate systems to understand and evaluate potential outcomes.  **Practice 6.1** Systematically test computational artifacts by considering all scenarios and using test cases.  **Practice 7.1** Select, organize, and interpret large data sets from multiple sources to support a claim. |
| 12. Understand the environmental, social, and economic impacts of decisions | **Practice 1.1** Include the unique perspectives of others and reflect on one's own perspectives when designing and developing computational products.  **Practice 1.2** Address the needs of diverse end users during the design process to produce artifacts with broad accessibility and usability.  **Practice 1.3** Employ self- and peer-advocacy to address bias in interactions, product design, and development methods. |

The CTE pathways also contain 11 knowledge and performance anchor standards that are common across each of the industry sectors:

1. Academics (sector-specific)
2. Communications
3. Career Planning and Management
4. Technology
5. Problem Solving and Critical Thinking
6. Health and Safety
7. Responsibility and Flexibility
8. Ethics and Legal Responsibilities
9. Leadership and Teamwork
10. Technical Knowledge and Skills
11. Demonstration and Application

Anchor standards 2, 4, 5, and 7-11 are all reinforced in the computer science content and practice standards.

### Information and Communication Technologies (ICT) Sector

*Note:* Specific CTE standards are referenced in parentheses following each CS standard.

Key: Null symbol (Ø) = not found

| **Information Support and Services Pathway Standard** | **Related 6–8 CS Standards** | **Related 9–12 Core CS Standards** | **Related 9–12 Specialty CS Standards** |
| --- | --- | --- | --- |
| A3.0 Access and transmit information in a networked environment. | 6-8.NI.4 Model the role of protocols in transmitting data across networks and the Internet (A3.1) | Ø | Ø |
| A5.0 Identify requirements for maintaining secure network systems. | 6-8.NI.5 Explain potential security threats and security measures to mitigate those threats. (A5.2, A5.3) | 9-12.NI.6 Compare and contrast security measures to address various security threats. (A5.2, A5.3) | 9-12Sp.NI.5 Develop solutions to security threats. (A5.2, A5.3) |
| A6.0 Diagnose and solve software, hardware, networking, and security problems. | 6-8.CS.3 Systematically apply troubleshooting strategies to identify and resolve hardware and software problems in computing systems. (A6.1, A6.2, A6.3) | 9-12.CS.3 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors. (A6.2, A6.3) | Ø |
| A8.0 Manage and implement information, technology, and communication projects. | 6-8.CS.2 Design a project that combines hardware and software components to collect and exchange data (A8.1, A8.5) | Ø | Ø |

| **Networking Pathway Standard** | **Related 6–8 CS Standards** | **Related 9–12 Core CS Standards** | **Related 9–12 Specialty CS Standards** |
| --- | --- | --- | --- |
| B1.0 Identify and describe the principles of networking and the technologies, models, and protocols used in a network. | 6-8.NI.4 Model the role of protocols in transmitting data across networks and the Internet. (B1.1, B1.3) | 9-12.NI.4 Describe issues that impact network functionality (B1.1, B1.3, B1.4) | 9-12Sp.NI.3 Examine the scalability and reliability of networks by describing the relationships between routers, switches, servers, topology, and addressing. (B1.1, B1.3, B1.4, B1.5) |
| B2.0 Identify, describe, and implement network media and physical topologies. | Ø | Ø | 9-12Sp.NI.3 Examine the scalability and reliability of networks by describing the relationships between routers, switches, servers, topology, and addressing. (B2.3) |
| B4.0 Demonstrate proper network administration and management skills. | Ø | 9-12.NI.4 Describe issues that impact network functionality. (B4.1) | Ø |
| B6.0 Use and assess network communication applications and infrastructure. | Ø | 9-12.NI.4 Describe issues that impact network functionality. (B4.1) | 9-12Sp.NI.3 Examine the scalability and reliability of networks by describing the relationships between routers, switches, servers, topology, and addressing. (B2.3) |
| B8.0 Identify security threats to a network and describe general methods to mitigate those threats. | 6-8.NI.5 Explain potential security threats and security measures to mitigate those threats (B8.1, B8.4, B8.5) | 9-12.NI.6 Compare and contrast security measures to address various security threats. (B8.1, B8.4, B8.5) | 9-12Sp.NI.5 Develop solutions to security threats. (B8.0) |

| **Software and Systems Development Pathway Standard** | **Related 6–8 CS Standards** | **Related 9–12 Core CS Standards** | **Related 9–12 Specialty CS Standards** |
| --- | --- | --- | --- |
| C1.0 Identify and apply the systems development process | 6-8.AP.18 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts. (C1.4, C1.5) | 9-12.AP.21 Design and develop computational artifacts working in team roles using collaborative tools. (C1.4) | 9-12S.AP.19 Plan and develop programs for broad audiences using a specific software life cycle process. (C1.1)  9-12S.AP.25 Use version control systems, integrated development environments, and collaborative tools and practices (code documentation) while developing software within a group. (C1.4, C1.5) |
| C2.0 Define and analyze systems and software requirements. | 6-8.AP.15 Seek and incorporate feedback from team members and users to refine a solution that meets user needs. (C2.3, C2.4, C2.5)  6-8.IC.21 Discuss issues of bias and accessibility in the design of existing technologies. (C2.2)  6-8.IC.22 Collaborate with many contributors when creating a computational artifact. (C2.3, C2.4, C2.5) | 9-12.AP.18 Systematically design programs for broad audiences by incorporating feedback from users (C2.3, C2.4, C2.5)  9-12.IC.24 Identify impacts of bias and equity deficit on design and implementation of computational artifacts and apply appropriate processes for evaluating issues of bias (C2.2) | 9-12S.IC.27 Evaluate computational artifacts with regard to improving their beneficial effects and reducing harmful effects on society. (C2.2) |
| C3.0 Create effective interfaces between humans and technology. | 6-8.CS.1 Design modifications to computing devices in order to improve the ways users interact with the devices. (C3.2, C3.3)  6-8.CS.2 Design a project that combines hardware and software components to collect and exchange data. (C3.1) | 9-12.CS.1 Describe ways in which abstractions hide the underlying implementation details of computing systems to simplify user experiences (P4.1)  9-12.IC.23 Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices (C3.3)  9-12.IC.24 Identify impacts of bias and equity deficit on design and implementation of computational artifacts and apply appropriate processes for evaluating issues of bias (C3.3) | Ø |
| C4.0 Develop software using programming languages. | 6-8.DA.7 Represent data in multiple ways. (C4.4)  6-8.AP.11 Create clearly named variables that store data, and perform operations on their contents. (C4.4)  6-8.AP.12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals. (C4.9)  6-8.AP.19 Document programs in order to make them easier to use, read, test, and debug. (C4.11) | 9-12.DA.8 Translate between different representations of data abstractions of real-world phenomena, such as characters, numbers, and images. (C4.4)  9-12.AP.13 Create more generalized computational solutions using collections instead of repeatedly using simple variables. (C4.7)  9-12.AP.14 Justify the selection of specific control structures by identifying tradeoffs associated with implementation, readability, and performance. (C4.9)  9-12.AP.15 Iteratively design and develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions. (C4.9)  9-12.AP.16 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or classes. (C4.8)  9-12.AP.17 Create computational artifacts using modular design. (C4.9)  9-12.AP.22 Document decisions made during the design process using text, graphics, presentations, and/or demonstrations in the development of complex programs. (C4.11) | 9-12S.AP.12 Implement searching and sorting algorithms to solve computational problems. (C4.10)  9-12S.AP.13 Evaluate algorithms in terms of their efficiency. (C4.10)  9-12S.AP.14 Compare and contrast fundamental data structures and their uses. (C4.7)  9-12S.AP.26 Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems. (C4.1) |
| C5.0 Test, debug, and improve software development work. | 6-8.AP.17 Systematically test and refine programs using a range of test cases. (C5.4, C5.5, C5.6) | 9-12.AP.20 Iteratively evaluate and refine a computational artifact to enhance its performance, reliability, usability, and accessibility. (C5.4, C5.5, C5.6) | 9-12S.AP.22 Develop and use a series of test cases to verify that a program performs according to its design specifications. (C5.4, C5.5, C5.6)  9-12Sp.AP.24 Evaluate key qualities of a program through a process such as code review. (C5.1) |
| C6.0 Integrate a variety of media into development projects. | 6-8.AP.16 Incorporate existing code, media, and libraries into original programs, and give attribution. (C6.6) | Ø | Ø |
| C8.0 Develop databases | 6-8.DA.8 Collect data using computational tools and transform the data to make it more useful (C8.7) | 9-12.DA.10 Create data visualizations to help others better understand real-world phenomena. (C8.8) | 9-12Sp.DA.7 Select and use data collection tools and techniques to generate data sets that reveal patterns or communicate information. (C8.8) |
| C9.0 Develop software for a variety of devices, including robotics. | Ø | Ø | 9-12Sp.AP.20 Develop programs for multiple computing platforms. (C9.0) |
| C10. Develop intelligent computing | Ø | Ø | 9-12.Sp.AP.10 Describe how artificial intelligence drives many software and physical systems. (C10.2)  9-12Sp.AP.11 Implement an algorithm that uses artificial intelligence to overcome a simple challenge (C10.4) |

| **Games and Simulation Pathway Standard** | **Related 6–8 CS Standards** | **Related 9–12 Core CS Standards** | **Related 9–12 Specialty CS Standards** |
| --- | --- | --- | --- |
| D3.0 Create a working game or simulation individually or as part of a team. | 6-8.AP.19 Document programs in order to make them easier to use, read, test, and debug (D3.2) | 9-12.AP.22 Document decisions made during the design process using text, graphics, presentations, and/or demonstrations in the development of complex programs (D3.2, D3.4) | Ø |
| D5.0 Integrate music, sound, art, and animation as it applies to the environmental design of the game/simulation. | 6-8.AP.16 Incorporate existing code, media, and libraries into original programs, and give attribution. (D5.1) | Ø | Ø |
| D6.0 Explain the role and principles of event modeling and interface design and apply those principles in a game/simulation design and project. | 6-8.DA.9 Test and analyze the effects of changing variables while using computational models. (D6.1, D6.2)  6-8.AP.15 Seek and incorporate feedback from team members and users to refine a solution that meets user needs. (D6.3) | 9-12.DA.11 Refine computational models to better represent the relationships among different elements or data collected from a phenomenon or process. (D6.1, D6.2)  9-12.AP.18 Systematically design programs for broad audiences by incorporating feedback from users. (D6.3)  9-12.AP.20 Iteratively evaluate and refine a computational artifact to enhance its performance, reliability, usability, and accessibility. (D6.3, D6.4) | Ø |
| D7.0 Acquire and apply appropriate programming skills for rendering a single player or multiuser game or simulation project, including program control, conditional branching, memory management, scorekeeping, timed event strategies, and implementation issues. | 6-8.NI.4 Model the role of protocols in transmitting data across networks and the Internet. (D7.1)  6-8.NI.5 Explain potential security threats and security measures to mitigate threats. (D7.1)  6-8.AP.17 Systematically test and refine programs using a range of test cases. (D7.3)  6-8.AP.18 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts. (D7.2)  6-8.AP.19 Document programs in order to make them easier to use, read, test, and debug. (D7.3, D7.4) | 9-12.NI.4 Describe issues that impact network functionality. (D7.1)  9-12.NI.6 Compare and contrast security measures to address various security threats. (D7.1)  9-12.AP.20 Iteratively evaluate and refine a computational artifact to enhance its performance, reliability, usability, and accessibility. (D7.3)  9-12.AP.22 Document decisions made during the design process using text, graphics, presentations, and/or demonstrations in the development of complex programs. (D7.2) | 9-12S.NI.3 Examine the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing. (D7.1)  9-12S.NI.5 Develop solutions to security threats. (D7.1)  9-12S.AP.20 Develop programs for multiple computing platforms. (D7.1)  9-12.S.21 Identify and fix security issues that might compromise computer programs. (D7.1)  9-12S.AP.22 Develop and use a series of test cases to verify that a program performs according to its design specifications. (D7.3)  9-12S.AP.24 Evaluate key qualities of a program through a process such as a code review. (D7.2)  9-12S.AP.25 Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (e.g., code documentation) while developing software within a group. (D7.2) |
| D8.0 Acquire and apply appropriate artificial intelligence (AI) techniques used by the game development industry. | Ø | Ø | 9-12.S.AP.10 Describe how artificial intelligence drives many software and physical systems. (D8.1)  9-12S.AP.11 Implement an algorithm that uses artificial intelligence to overcome a simple challenge (D8.2) |

### Other Sectors

*Note:* Specific CTE standards are referenced in parentheses following each CS standard.

Key: Null symbol (Ø) = not found

| **Arts, Media, and Entertainment: Game Design and Integration Pathway Standard** | **Related 6–8 CS Standards** | **Related 9–12 Core CS Standards** | **Related 9–12 Specialty CS Standards** |
| --- | --- | --- | --- |
| D2.0 Analyze the core tasks and challenges of video game design and explore the methods used to create and sustain player immersion. | 6-8.AP.13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. (D2.2) | 9-12.AP.16 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or classes. (D2.2) | 9-12S.AP.16 Analyze a large-scale computational problem and identify generalizable patterns or problem components that can be applied to a solution. (D2.2) |
| D3.0 Acquire and apply appropriate game programming concepts and skills to develop a playable video game. | 6-8.AP.11 Create clearly named variables that store data, and perform operations on their contents. (D3.1)  6-8.AP.12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals. (D3.1) | 9-12.DA.8 Translate between different representations of data abstractions of real-world phenomena, such as characters, numbers, and images. (D3.1)  9-12.AP.13 Create more generalized computational solutions using collections instead of repeatedly using simple variables. (D3.1)  9-12.AP.14 Justify the selection of specific control structures by identifying tradeoffs associated with implementation, readability, and performance. (D3.1)  9-12.AP.15 Iteratively design and develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions. (D3.1) | 9-12S.AP.14 Compare and contrast fundamental data structures and their uses. (D3.1) |
| D4.0 Students will demonstrate mastery of game art and multimedia, including music, sound, art, and animation. | 6-8.AP.16 Incorporate existing code, media, and libraries into original programs, and give attribution. (D4.8)  6-8.IC.23 Compare tradeoffs associated with licenses for computational artifacts to balance the protection of the creators' rights and the ability for others to use and modify. (D4.8) | 9-12.AP.19 Explain the limitations of licenses that restrict use of computational artifacts when using resources such as libraries. (D4.8)  9-12.IC.27 Explain the beneficial and harmful effects that intellectual property laws can have on innovation. (D4.8) | 9-12S.IC.30 Debate laws and regulations that impact the development and use of software. (D4.8) |
| D5.0 Demonstrate an understanding of testing techniques used to evaluate, assess, rate, and review quality assurance of video games. | 6-8.AP.17 Systematically test and refine programs using a range of test cases. (D5.2)  6-8.IC.21 Discuss issues of bias and accessibility in the design of existing technologies. (D5.4) | 9-12.AP.20 Iteratively evaluate and refine a computational artifact to enhance its performance, reliability, usability, and accessibility. (D5.2) | 9-12S.AP.22 Develop and use a series of test cases to verify that a program performs according to its design specifications. (D5.2) |
| D6.0 Understand the general procedures, documentation, and requirements of large scale game design projects. Examine and categorize the significant processes in the production of games. | 6-8.AP.18 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts. (D6.4)  6-8.AP.19 Document programs in order to make them easier to use, read, test, and debug. (D6.7) | 9-12.AP.21 Design and develop computational artifacts working in team roles using collaborative tools. (D6.1, D6.4)  9-12.AP.22 Document decisions made during the design process using text, graphics, presentations, and/or demonstrations in the development of complex programs. (D6.7) | 9-12S.AP.25 Use version control systems, integrated development environments, and collaborative tools and practices (code documentation) while developing software within a group. (D6.1, D6.4) |
| D10.0 Students will build a game that demonstrates teamwork and project management by creating a game design production plan that describes the game play, outcomes, controls, rewards, interface, and artistic style of a video game. | 6-8.AP.15 Seek and incorporate feedback from team members and users to refine a solution that meets user needs. (D10.2)  6-8.AP.17 Systematically test and refine programs using a range of test cases. (D10.6)  6-8.AP.18 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts. (D10.0)  6-8.AP.19 Document programs in order to make them easier to use, read, test, and debug. (D10.1, 10.6) | 9-12.AP.18 Systematically design programs for broad audiences by incorporating feedback from users (D10.2)  9-12.AP.20 Iteratively evaluate and refine a computational artifact to enhance its performance, reliability, usability, and accessibility. (D10.6)  9-12.AP.21 Design and develop computational artifacts working in team roles using collaborative tools. (D10.0)  9-12.AP.22 Document decisions made during the design process using text, graphics, presentations, and/or demonstrations in the development of complex programs. (D10.1) | 9-12S.AP.22 Develop and use a series of test cases to verify that a program performs according to its design specifications. (D10.6)  9-12S.AP.25 Use version control systems, integrated development environments, and collaborative tools and practices (code documentation) while developing software within a group. (D10.0) |

| **Energy, Environment, and Utilities: Telecommunications Pathway Standard** | **Related 6–8 CS Standards** | **Related 9–12 Core CS Standards** | **Related 9–12 Specialty CS Standards** |
| --- | --- | --- | --- |
| C1.0 Understand the basic principles and concepts that impact the telecommunications industry, including systems, concepts, and regulations. | 6-8.IC.20 Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options. (C1.1, C1.2) | 9-12.NI.5 Distinguish design characteristics of the Internet from other networks. (C1.1) | 9-12.NI.6 Explain how the characteristics of the Internet influence the systems developed on it. (C1.1) |
| C2.0 Demonstrate understanding and use of the basic and emerging technologies that impact the telecommunications industry. | 6-8.NI.4 Model the role of protocols in transmitting data across networks and the Internet. (C2.9) | Ø | Ø |
| C3.0 Examine the role and functions of satellites in telecommunications. | 6-8.NI.4 Model the role of protocols in transmitting data across networks and the Internet. (C3.3, C3.8) | 9-12.NI.4 Describe issues that impact network functionality. (C3.3, C3.8) | 9-12S.NI.3 Explain the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing. (C3.3, C3.8) |
| C4.0 Research the components, interaction, and operations of wireless telecommunications systems. | 6-8.NI.4 Model the role of protocols in transmitting data across networks and the Internet. (C4.4) | 9-12.NI.4 Describe issues that impact network functionality. (C4.4) | 9-12S.NI.3 Explain the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing. (C4.4) |
| C5.0 Research the components, interaction, and operations of fixed-wire telecommunications systems. | 6-8.NI.4 Model the role of protocols in transmitting data across networks and the Internet. (C5.3) | 9-12.NI.4 Describe issues that impact network functionality. (C5.3) | 9-12S.NI.3 Explain the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing. (C5.3) |
| C6.0 Consider privacy and security issues of the telecommunications systems. | 6-8.NI.5 Explain potential security threats and security measures to mitigate those threats. (C6.2)  6-8.IC.24 Compare tradeoffs between allowing information to be public and keeping information private and secure. (C6.2) | 9-12.NI.6 Compare and contrast security measures to address various security threats. (C6.20) | 9-12S.NI.3 Examine the scalability and reliability of networks by describing the relationships between routers, switches, servers, topology, and addressing. (C6.1) |

| **Public Services Public Safety Pathway Standard** | **Related 6–8 CS Standards** | **Related 9–12 Core CS Standards** | **Related 9–12 Specialty CS Standards** |
| --- | --- | --- | --- |
| A8.0 Demonstrate an understanding of the functions and career opportunities within the U.S. Department of Homeland security (DHS). | 6-8.NI.5 Explain potential security threats and security measures to mitigate threats. (A8.5)  6-8.NI.6 Apply multiple methods of information protection to model the secure transmission of information. (A8.5) | 9-12.NI.6 Compare and contrast security measures to address various security threats. (A8.4)  9-12.NI.7 Compare and contrast cryptographic techniques to model the secure transmission of information. A8.4)  9-12.IC.28 Explain the privacy concerns related to the collection and generation of data through automated processes. (A8.3)  9-12.IC.29 Evaluate the social and economic implications of privacy in the context of safety, law, or ethics. (A8.3) | 9-12S.NI.5 Develop solutions to security threats. (A8.6)  9-12S.NI.6 Analyze cryptographic techniques to model the secure transmission of information. (A8.6) |

| **Health Science and Medical Technology Pathway Standard** | **Related 6–8 CS Standards** | **Related 9–12 Core CS Standards** | **Related 9–12 Specialty CS Standards** |
| --- | --- | --- | --- |
| A5.0 Integrate computer skills into program components. | 6-8.DA.8 Collect data using computational tools and transform the data to make it more useful. (A5.2) | 9-12.DA.10 Create data visualizations to help others better understand real-world phenomena. (A5.2) | 9-12S.DA.7 Select and use data collection tools and techniques to generate data sets. (A5.2) |

## Connections to Postsecondary Education

### California State University/University of California Freshman Minimum Admission Requirements

The California State University (CSU) and University of California (UC) systems currently accept some computer science courses to fulfill freshman minimum admission requirements in category “c” (mathematics), “d” (laboratory science), or “g” (college preparatory elective). Alignments between the CA computer science standards and the CA mathematics and science standards and practices are described in detail in Section IV: Interdisciplinary Connections.

For a computer science course to meet the requirements for category “c”, it must be designed to give students five core competencies:

1. A view that mathematics is not just a collection of definitions, algorithms and/or theorems to memorize and apply, but rather is a coherent and tightly organized body of knowledge that provides a way to think about and understand a broad array of phenomena.
2. A proclivity to put time and thought into using mathematics to grasp and solve unfamiliar problems that may not match examples the student has seen before.
3. A view that mathematics models reality and students should have the capacity to use mathematical models to guide their understanding of the world around us.
4. An awareness of special goals of mathematics, such as clarity and brevity, parsimony, universality and objectivity.
5. Confidence and fluency in handling formulas and computational algorithms: understanding their motivation and design, predicting approximate outcomes and computing them – mentally, on paper or with technology, as appropriate.

For more information on category “c” course requirements, see <http://www.ucop.edu/agguide/a-g-requirements/c-mathematics/index.html>.

For a computer science course to meet the requirements for category “d”, it must align with the eight practices of science and engineering identified in the California Next Generation Science Standards. For more information on category “d” course requirements, see <http://www.ucop.edu/agguide/a-g-requirements/d-lab-science/index.html>.

Students and parents can search the University of California’s “a-g” approved course list (<http://www.ucop.edu/agguide/>) to determine if a high school’s computer science course satisfies any of the minimum admission requirements. School administrators can submit computer science courses for approval through the University of California’s “a-g” course management portal (https://hs-articulation.ucop.edu/agcmp/login#/).

### Advanced Placement (AP)

Students who receive a score of 3 or higher on any Advanced Placement (AP) computer science exam can receive credit for a category G course. Currently, there are two AP computer science exams.

AP Computer Science Principles (CSP) is a newer course that covers the fundamental principles of computer science. It is equivalent to a first-semester introductory college computer science course. AP CSP emphasizes computational thinking skills and computer science practices similar to those in the California computer science standards. The course covers seven “Big Ideas” in computing: creativity, abstraction, data and information, algorithms, programming, the internet, and global impacts. There is significant overlap between the learning objectives of AP CSP and the CA computer science standards for 9–12 (core and specialty). The CA computer science standards contain a few additional standards that cover advanced topics in networking and computer programming. For more information about AP CSP, please visit the College Board web site at <https://apstudent.collegeboard.org/apcourse/ap-computer-science-principles/course-details>.

AP Computer Science A (CS A) is a more traditional course that covers the fundamentals of Java programming. It is an introductory course for all students interested in the fundamentals of programming, not just those who intend to major in computer science in college. The topic outline for AP CS A contains six major topics: object-oriented program design, program implementation, program analysis, standard data structures, standard operations and algorithms, and computing in context. The AP CS A course outline does not include topics that are included in the Networks and the Internet and Data and Analysis concept areas of the CA CS standards. In addition practices like inclusion, collaboration, and communicating about computing are de-emphasized in AP CS A. For more information about AP CS A, please visit the College Board web site at <https://apstudent.collegeboard.org/apcourse/ap-computer-science-a/course-details>.

Neither the AP CSP course nor the AP CS A course cover all the 9–12 core CS standards. The table below describes the alignment between the content of the AP computer science courses and the CA CS standards for 9–12. Both AP courses contain more specific learning goals than those outlined here; this table provides a general overview of areas where there is the greatest overlap.

Key: Null symbol (Ø) = not found

| CA CS Standard Identifier | **CA CS Standard** | **AP CSP Learning Objectives (alignments to Essential Knowledge are given by the letters in parentheses)** | **AP CS A Topic Outline** |
| --- | --- | --- | --- |
| 9-12.CS.1 | Describe ways in which abstractions hide the underlying implementation details of computing systems to simplify user experiences. | LO 2.1.1 Describe the variety of abstractions used to represent data (A, B, C, D, E, F, G).  LO 2.2.3 Identify multiple levels of abstractions that are used when writing programs (A, B, C, D, E, F, G, H, I, J, K).  LO 6.1.1 Explain the abstractions in the Internet and how the Internet functions (A, B, C, D, E, F, G, H, I) | IV C. Classes |
| 9-12.CS.2 | Compare levels of abstraction and interactions between application software, system software, and hardware. | LO 2.2.2 Use multiple levels of abstraction to write programs (A, B).  LO 2.2.3 Identify multiple levels of abstractions that are used when writing programs (B, C, D, E, F, G, H, I, J, K). | Ø |
| 9-12.CS.3 | Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors. | LO 5.4.1 Evaluate the correctness of a program (E). | III. B. Debugging |
| 9-12.NI.4 | Describe issues that impact network functionality. | LO 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it (A, B, C, D, E, F). | Ø |
| 9-12.NI.5 | Distinguish design characteristics of the Internet from other networks. | LO 6.1.1 Explain the abstractions in the Internet and how the Internet functions (A, B, C, D, F).  LO 6.2.1 Explain the characteristics of the Internet and the systems built on it (A, D). | Ø |
| 9-12.NI.6 | Compare and contrast security measures to address various security threats. | LO 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it (C, D, E, F, G, H). | VI. D. Social and ethical ramifications of computer use |
| 9-12.NI.7 | Compare and apply encryption techniques to model the secure transmission of information. | LO 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it (H, I, J, K, L). | Ø |
| 9-12.DA.8 | Translate between different representations of data abstractions of real-world phenomena, such as characters, numbers, and images. | LO 2.1.1 Describe the variety of abstractions used to represent data. (A, B, C)  LO 2.1.2 Explain how binary sequences are used to represent digital data (D, E, F). | Ø |
| 9-12.DA.9 | Describe tradeoffs associated with how data elements are organized and stored. | LO 3.1.1 Find patterns and test hypotheses about digitally processed information to gain insight and knowledge (B, C).  LO 3.3.1 Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information (C, G, H). | V. A. Operations on data structures |
| 9-12.DA.10 | Create data visualizations to help others better understand real-world phenomena. | LO 3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notations, and precise language (A, B, C, D, E). | Ø |
| 9-12.DA.11 | Refine computational models to better represent the relationships among different elements of data collected from a phenomenon or process. | LO 2.3.1 Use models and simulations to represent phenomena (D).  LO 2.3.2 Use models and simulations to formulate, refine, and test hypotheses (A). | Ø |
| 9-12.AP.12 | Design algorithms to solve computational problems using a combination of original and existing algorithms. | LO 4.1.1 Develop and algorithm for implementation in a program (E, F, G, H). | V. A. Operations on data structures  V. B. Searching  V. C. Sorting |
| 9-12.AP.13 | Create more generalized computational solutions using collections instead of repeatedly using simple variables. | LO 5.5.1 Employ appropriate mathematical and logical concepts in programming (H, I, J). | V. A. Operations on data structures  V. B. Searching  V. C. Sorting  IV. D. Lists  IV. E. Arrays |
| 9-12.AP.14 | Justify the selection of specific control structures by identifying tradeoffs associated with implementation, readability, and performance. | LO 5.4.1 Evaluate the correctness of a program (I, J).  LO 5.5.1 Employ appropriate mathematical and logical concepts in programming. (D, E, F, G, I) | II. B. Programming constructs |
| 9-12.AP.15 | Iteratively design and develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions. | LO 1.2.1 Create a computational artifact for creative expression (E).  LO 5.1.1 Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge (A, B).  LO 5.1.2 Develop a correct program to solve problems (A). | II. A. Implementation techniques |
| 9-12.AP.16 | Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or classes. | LO 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. (A, B, C)  LO 2.2.2 Use multiple levels of abstraction to write programs. (A–B)  LO 5.3.1 Use abstraction to manage complexity in programs (A–G, L). | I. A. Program and class design  II. A. Implementation techniques  IV. C. Classes |
| 9-12.AP.17 | Create computational artifacts using modular design. | LO 1.2.1 Create a computational artifact for creative expression (B).  LO 1.2.3 Create a new computational artifact by combining or modifying existing artifacts (A).  LO 5.1.2 Develop a correct program to solve problems (B, C). | I. A. Program and class design  II. A. Implementation techniques |
| 9-12.AP.18 | Systematically design programs for broad audiences by incorporating feedback from users. | LO 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts (A–D).  LO 5.1.2 Develop a correct program to solve problems (G, H). | Ø |
| 9-12.AP.19 | Explain the limitations of licenses that restrict use of computational artifacts when using resources such as libraries. | LO 7.3.1 Analyze the beneficial and harmful impacts of computing (N–Q) | VI. C. Legal issues and intellectual property |
| 9-12.AP.20 | Iteratively evaluate and refine a computational artifact to enhance its performance, reliability, usability, and accessibility. | LO 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts (A–D).  LO 5.1.2 Develop a correct program to solve problems (A, H). | III. A. Testing  III. B. Debugging |
| 9-12.AP.21 | Design and develop computational artifacts working in team roles using collaborative tools. | LO 1.2.4 Collaborate in the creation of computational artifacts (A, B).  LO 5.1.3 Collaborate to develop a program (C). | Ø |
| 9-12.AP.22 | Document decisions made during the design process using text, graphics, presentations, and/or demonstrations in the development of complex programs. | LO 5.1.2 Develop a correct program to solve problems (D–F). | Ø |
| 9-12.IC.23 | Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices. | LO 7.3.1 Analyze the beneficial and harmful impacts of computing (A–Q) | VI. D. Social and ethical ramifications of computer use |
| 9-12.IC.24 | Identify impacts of bias and equity deficit on design and implementation of computational artifacts and apply appropriate processes for evaluating issues of bias. | LO 7.4.1 Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts. (A, C, D) | VI. D. Social and ethical ramifications of computer use |
| 9-12.IC.25 | Demonstrate ways a given algorithm applies to problems across disciplines. | LO 5.2.1 Explain how programs implement algorithms (J). | Ø |
| 9-12.IC.26 | Use collaboration tools and methods to increase connectivity with people of different cultures and careers. | LO 1.2.4 Collaborate in the creation of computational artifacts (C, E).  LO 5.1.3 Collaborate to develop a program (B, C, F). | Ø |
| 9-12.IC.27 | Explain the beneficial and harmful effects that intellectual property laws can have on innovation. | LO 7.3.1 Analyze the beneficial and harmful effects of computing. (N, O, P, Q) | VI. C. Legal issues and intellectual property |
| 9-12.IC.28 | Explain the privacy concerns related to the collection and generation of data through automated processes. | LO 3.2.2 Determine how large data sets impact the use of computational processes to discover information and knowledge (D).  LO 3.3.1 Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information (A, B, F). | VI. B. Privacy |
| 9-12.IC.29 | Evaluate the social and economic implications of privacy in the context of safety, law, or ethics. | LO 7.3.1 Analyze the beneficial and harmful impacts of computing (G, H) | VI. B. Privacy  VI. D. Social and ethical ramifications of computer use |
| 9-12S.CS.1 | Illustrate ways computing systems implement logic, input, and output through hardware components. | LO 2.2.3 Identify multiple levels of abstractions that are used when writing programs (E–I). | Ø |
| 9-12S.CS.2 | Categorize and describe the different functions of operating system software. | Ø | Ø |
| 9-12S.NI.3 | Examine the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing. | LO 6.2.2 Explain how the characteristics of the Internet influence the systems built on it. (A, B) | Ø |
| 9-12S.NI.4 | Explain how the characteristics of the Internet influence the systems developed on it. | LO 6.1.1 Explain the abstractions in the Internet and how the Internet functions (A–D, F).  LO 6.2.1 Explain the characteristics of the Internet and the systems built on it (A, D).  LO 6.2.2 Explain how the characteristics of the Internet influence the systems built on it (A–K). | Ø |
| 9-12S.NI.5 | Develop solutions to security threats. | LO 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it (C, G). | Ø |
| 9-12S.NI.6 | Analyze and apply encryption techniques to model the secure transmission of information. | LO 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it (I, K, L). | Ø |
| 9-12S.DA.8 | Use data analysis tools and techniques to identify patterns in data representing complex systems. | LO 3.2.1 Extract information from data to discover and explain connections or trends (C–F). | Ø |
| 9-12S.DA.7 | Select and use data collection tools and techniques to generate data sets that reveal patterns or communicate information. | LO 3.2.2 Determine how large data sets impact the use of computational processes to discover information and knowledge (B, C). | Ø |
| 9-12S.DA.9 | Evaluate the ability of models and simulations to test and support the refinement of hypotheses. | LO 2.3.2 Use models and simulations to formulate, refine, and test hypotheses (A–H). | Ø |
| 9-12S.AP.10 | Describe how artificial intelligence drives many software and physical systems. | Ø | Ø |
| 9-12S.AP.11 | Implement an algorithm that uses artificial intelligence to overcome a simple challenge. | Ø | Ø |
| 9-12S.AP.12 | Implement searching and sorting algorithms to solve computational problems. | LO 4.1.1 Develop and algorithm for implementation in a program (A–C).  LO 4.1.2 Express an algorithm in a language (G).  LO 4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity (H). | V. B. Searching  V. C. Sorting |
| 9-12S.AP.13 | Evaluate algorithms in terms of their efficiency. | LO 4.2.1 Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time (B, C).  LO 4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity (D, G). | III. E. Algorithm analysis |
| 9-12S.AP.14 | Compare and contrast fundamental data structures and their uses. | LO 5.3.1 Use abstraction to manage complexity in programs (K, L). | IV. D. Lists  IV. E. Arrays |
| 9-12S.AP.15 | Demonstrate the flow of execution of a recursive algorithm. | Ø | II. B. Programming constructs |
| 9-12S.AP.16 | Analyze a large-scale computational problem and identify generalizable patterns or problem components that can be applied to a solution. | LO 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts (A, B, C). | Ø |
| 9-12S.AP.17 | Construct solutions to problems using student-created components, such as procedures, modules, and/or objects. | LO 5.3.1 Use abstraction to manage complexity in programs (A–G, L). | I. A. Program and class design |
| 9-12S.AP.18 | Demonstrate code reuse by creating programming solutions using libraries and APIs. | LO 5.3.1 Use abstraction to manage complexity in programs (M, N, O). | I. A. Program and class design |
| 9-12S.AP.19 | Plan and develop programs for broad audiences using a specific software life cycle process. | LO 5.1.2 Develop a correct program to solve problems (A) | Ø |
| 9-12S.AP.20 | Develop programs for multiple computing platforms. | Ø | Ø |
| 9-12S.AP.21 | Identify and fix security issues that might compromise computer programs. | Ø | Ø |
| 9-12S.AP.22 | Develop and use a series of test cases to verify that a program performs according to its design specifications. | LO 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts (A, B). | III A. Testing |
| 9-12S.AP.23 | Modify an existing program to add additional functionality and discuss intended and unintended implications. | LO 5.1.2 Develop a correct program to solve problems (B, C). | Ø |
| 9-12S.AP.24 | Evaluate key qualities of a program through a process such as a code review. | LO 5.4.1 Evaluate the correctness of a program (A, B, C). | Ø |
| 9-12S.AP.25 | Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (e.g., code documentation) while developing software within a group. | LO 1.2.1 Create a computational artifact for creative expression (C).  LO 1.2.4 Collaborate in the creation of computational artifacts (A, B).  LO 5.1.3 Collaborate to develop a program (C, D, E, F.) | Ø |
| 9-12S.AP.26 | Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems. | LO 2.2.3 Identify multiple levels of abstractions that are used when writing programs (A, B). | Ø |
| 9-12S.IC.27 | Evaluate computational artifacts with regard to improving their beneficial effects and reducing harmful effects on society. | LO 5.1.1 Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge (E, F). | VI. D. Social and ethical ramifications of computer use |
| 9-12S.IC.28 | Evaluate how computational innovations that have revolutionized aspects of our culture might evolve. | LO 7.1.1 Explain how computing innovations affect communication, interaction, and cognition (A, B, C, D, E, F, G, H, I, J, K, L, M, N, O). | VI. D. Social and ethical ramifications of computer use |
| 9-12S.IC.29 | Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society. | LO 7.4.1 Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts (A–E). | VI. D. Social and ethical ramifications of computer use |
| 9-12S.IC.30 | Debate laws and regulations that impact the development and use of software. | LO 7.3.1 Analyze the beneficial and harmful impacts of computing (N–Q) | VI. C. Legal issues and intellectual property |

### International Baccalaureate (IB)

Computer science is considered an experimental science in the International Baccalaureate (IB) Diploma Programme curriculum, alongside biology, chemistry, design technology, physics, and environmental systems and societies. Computer science is offered as a standard level (SL) course, intended for students with no previous experience with computer science, and as a higher level (HL) course, intended for students with some prior exposure to programming. Both the SL and the HL course emphasize computational thinking and include the following four topics:

1. System fundamentals
2. Computer organization
3. Networks
4. Computational thinking, problem-solving, and programming

Students at SL and HL will also choose to study one of the following options:

1. Databases
2. Modeling and simulation
3. Web science
4. Object-oriented programming

The HL course includes three additional topics as well as more in-depth content for the selected option:

1. Abstract data structures
2. Resource management
3. Control

There is significant overlap between the IB Diploma Programme computer science courses and the CA CS standards, but neither the SL nor the HL course addresses the full set of CA CS standards. The IB courses include some objectives that are covered in the K–8 CS standards and explore topics such as system fundamentals and computer organization in much greater depth. The IB syllabus changes frequently -- teachers should attend an IB workshop to obtain detailed curriculum information. A general overview of the degree of alignment between the IB course topics and the CA CS standards is provided in the table below.

Key: Null symbol (Ø) = not found

| **IB CS Topic** | **Degree of Alignment with CA CS Standards** | **Example 9–12 CA CS Standards Aligned to CS Topic** |
| --- | --- | --- |
| Topic 1: System fundamentals | Low/None | Ø |
| Topic 2: Computer organization | Moderate | 9-12.CS.2, 9-12S.CS.1, 9-12S.CS.2 |
| Topic 3: Networks | Moderate | 9-12.NI.4, 9-12S.NI.3 |
| Topic 4: Computational thinking, problem-solving, and programming | Moderate | 9-12.AP.12, 9-12S.AP.10-13 |
| Topic 5: Abstract data structures | High | 9-12.AP.13, 9-12S.AP.14, 9-12S.AP.15 |
| Topic 6: Resource management | Low/None | Ø |
| Topic 7: Control | Low/None | Ø |
| Option A: Databases | Low/None | Ø |
| Option B: Modelling and simulation | Moderate | 9-12.DA.10, 9-12S.DA.7, 9-12S.DA.8 |
| Option C: Web science | Moderate | 9-12.NI.5, 9-12S.NI.4 |
| Option D: Object-oriented programming | High | 9-12.AP.16, 9-12.AP.17, 9-12S.AP.16, 9-12S.AP.17, 9-12S.AP.18 |
| Case study | Low/None | Ø |

## Glossary

The glossary includes definitions of terms used in the statements in the framework. These terms are defined for readers of the framework and are not necessarily intended to be the definitions or terms that are seen by students. *Sources for definitions of terms are located in the References after the Glossary.*

| **Term** | **Definition** |
| --- | --- |
| **abstraction** | *(process):* The process of hiding detail associated with an idea or phenomenon to reduce complexity and facilitate communication. By hiding some details, abstraction allows one to focus on relevant details without including background details or explanations.  *(product):* A representation of an idea or phenomenon that hides details irrelevant to the question at hand. Abstractions can be associated with levels, where abstractions that are used to define other abstractions are commonly referred to as lower level abstractions, and abstractions built upon lower level abstractions are considered higher level abstractions. |
| **accessibility** | The extent to which products, devices, services, or environments can be used by people who experience disabilities. Accessibility standards that are generally accepted by professional groups include the Web Content Accessibility Guidelines (WCAG) 2.0 and Accessible Rich Internet Applications (ARIA) standards. [Source: Wikipedia] |
| **algorithm** | A sequence of instructions designed to complete a specific task. |
| **analog** | The defining characteristic of data that is represented in a continuous, physical way. Whereas digital data is represented in discrete binary form, analog data uses continuous representations such as the surface grooves on a vinyl record, the magnetic tape of a VCR cassette, or electromagnetic waves sent to an analog speaker. [Source: Techopedia] |
| **app** | An app is computer software, or a program, most commonly a small, specific one used for mobile devices. The term app originally referred to any mobile or desktop application, but as more app stores have emerged to sell mobile apps to smartphone and tablet users, the term has evolved to refer to small programs that can be downloaded and installed all at once. Also known as a *mobile application*.  [Source: Techopedia] |
| **Application Programming Interface (API)** | A set of commands, procedures, protocols, and objects that programmers can use to create or interact with software. A good API makes it easier to develop a computer program by providing building blocks for common operations, which are then put together by the programmer. Documentation for the API is usually provided to facilitate usage. [Sources: Tech Terms; Wikipedia] |
| **argument** | A value passed to a procedure and stored in a parameter.  *See also* parameter. |
| **array** | Arrays are commonly used in computer programs to organize data in a sequence, where individual data values can be easily accessed and/or modified. Each element in an array is identified by an index. |
| **artifact** | *See the definition for* computational artifact*.* |
| **audience** | Expected end users of a computational artifact or system.  *See also* end user. |
| **authentication** | The verification of the identity of a person or process.  [Source: Free On-Line Dictionary Of Computing] |
| **automate; automation** | **automate:** To perform a process or procedure without human assistance. [Source: Wikipedia]  **automation**: The process of automating. |
| **bandwidth** | The maximum data transfer rate of a network or Internet connection. It measures how much data can be sent over a specific connection in a given amount of time.  [Source: Tech Terms] |
| **binary** | A numeric system that only uses two digits -- 0 and 1. Computers operate in binary, meaning data is represented and calculations are performed using only zeros and ones. Binary is also called “base 2”. [Source: Tech Terms] |
| **biometric verification** | An authentication process used to confirm a claimed identity through uniquely identifiable biological traits, such as fingerprints and hand geometry. It is designed to allow a user to prove his or her identity by supplying a biometric sample and associated unique identification code in order to gain access to a secure environment. [Source: Techopedia] |
| **Boolean** | A type of data or expression with two possible values: *true* and *false.* [Source: Free On-Line Dictionary of Computing] |
| **bug** | An error in a software program. It may cause a program to unexpectedly quit or behave in an unintended manner. The process of finding and correcting errors (bugs) is called debugging. [Sources: Tech Terms; Wikipedia]  *See also* debugging. |
| **camel case** | Camel case (also "CamelCase" or "dromedary case") is a naming convention in which the first letter of each word in a compound word is capitalized. Some programming languages do not allow the use of spaces in the names of procedures, variables, or other entities. Therefore, programmers often use CamelCase to define portions of their code. For example, employeeID, employeeFirstName, employeeLastName, and employeeAddress. [Sources: Tech Terms] |
| **cloud; cloud computing** | Cloud computing is a type of Internet-based computing that relies on sharing computer resources. Instead of installing software applications on local servers or devices, the applications and services are offered over the Internet, from data centers all over the world. These data centers are collectively referred to as the “cloud.” [Sources: Tech Terms; Webopedia] |
| **code** | **code** *(n)*: Any set of instructions expressed in a programming language. [Source: Massachusetts Department of Elementary and Secondary Education, 2016]  **code** *(v)*: To write instructions for a computer using a programming language. *See also* program; programming. |
| **collection** | A set of variables used to store and process related data. The data is accessible through the use of a single variable and functionality defined for that particular collection. Examples of collections include arrays, sets, and lists. |
| **comment** | A programmer-readable annotation in the code of a computer program added to make the code easier to understand. Comments are generally ignored by machines. [Source: Wikipedia] |
| **complexity** | The minimum amount of resources, such as memory, time, or messages, needed to solve a problem or execute an algorithm. [Source National Institute of Standards and Technology –Dictionary of Algorithms and Data Structures] |
| **component** | Computers are made up of many different parts, or components, such as a motherboard and hard drive. Each of these parts is made up of smaller parts, also called components. For example, a motherboard includes a circuit board, electrical connectors, and resistors. These and other components work together to make the motherboard function. [Source: Tech Terms] |
| **compose** | To make or create by putting together parts of elements. [Source: Free Dictionary]  *See also* decompose. |
| **compound conditional** | Compound conditionals combine two or more conditions in a logical relationship (e.g., using AND, OR, and NOT).  *See also* conditional. |
| **computational** | Relating to computers or computing methods. |
| **computational artifact** | Anything created by a human using a computational thinking process and a computing device. A computational artifact can be, but is not limited to, a program, image, audio, video, presentation, or web page file. [Source: College Board, 2016] |
| **computational model;**  **computational modeling** | Computational modeling is the use of computers to simulate and study the behavior of complex systems. A computational model contains numerous variables that characterize the system being studied. Simulation is done by adjusting each of these variables alone or in combination and observing how the changes affect the outcomes. The results of model simulations help researchers make predictions about what will happen in the real system that is being studied in response to changing conditions. [Source: National Institute of Biomedical Imaging and Bioengineering, 2016] |
| **computational thinking** | The human ability to formulate problems so that their solutions can be represented as computational steps or algorithms to be executed by a computer. Computational thinking plays a key role in the computer science practices described in the K-12 Computer Science Framework and encompasses practices #3: Recognizing and Defining Computational Problems, #4: Developing and Using Abstractions, #5: Creating Computational Artifacts, and #6: Testing and Refining Computational Artifacts. [Source: Lee, 2016; K12 CS Framework] |
| **computer** | A machine or device that performs processes, calculations, and operations based on instructions provided by a software or hardware program. [Source: Techopedia]  *See also* computing device. |
| **computer science** | The study of computers and algorithmic processes, including their principles, their hardware and software designs, their implementation, and their impact on society. [Source: Association for Computing Machinery, 2006] |
| **computing** | Any goal-oriented activity requiring, benefiting from, or creating computers. Computing is a family of disciplines that includes computer science, electrical engineering, and information systems. [Sources: Association for Computing Machinery; Institute of Electrical and Electronic Engineers, 2006] |
| **computing device** | A physical device that uses hardware and software to receive, process, and output information. Computers, mobile phones, and computer chips inside appliances are all examples of computing devices. |
| **computing system** | A collection of one or more computers or computing devices, together with their hardware and software, integrated for the purpose of accomplishing shared tasks. Although a computing system can be limited to a single computer or computing device, it more commonly refers to a collection of multiple connected computers, computing devices, and hardware. |
| **conditional** | A programming language feature that determines the flow of control of a program. A conditional can appear in the form of a conditional statement (if - then), conditional expression (Boolean expression), or conditional construct (functional programming). |
| **configuration** | *(process):* Defining the options that are provided when installing or modifying hardware and software or the process of creating the configuration (product). [Source: TechTarget]  *(product):* The specific hardware and software details that tell exactly what the system is made up of, especially in terms of devices attached, capacity, or capability. [Source: TechTarget] |
| **connection** | A physical or wireless attachment between multiple computing systems, computers, or computing devices. |
| **connectivity** | A [program](http://www.webopedia.com/TERM/P/program.html)’s or device’s ability to link with other programs and devices. [Source: Webopedia] |
| **control; control structure** | **control:** (*in programming*) The use of elements of programming code to direct which actions take place and the order in which they take place.  **control structure:** A programming (code) structure that implements control. Conditionals and loops are examples of control structures. |
| **Creative Commons license** | One of several public copyright licenses that enable the free distribution of an otherwise copyrighted work. A Creative Commons (CC) license is used when an author wants to give people the right to share, use, and build upon a work that they have created. A CC license provides an author flexibility (for example, they might choose to allow only non-commercial uses of their own work) and protects the people who use or redistribute an author's work from concerns of copyright infringement as long as they abide by the conditions that are specified in the license by which the author distributes the work. [Source: Wikipedia]  *See also* license. |
| **cryptography** | Cryptography is a technique for transforming information on a computer in such a way that it becomes unreadable by anyone except authorized parties. Cryptography is useful for supporting secure communication of data across networks. Examples of cryptographic methods include hashing, symmetric encryption/decryption (private key), and asymmetric encryption/decryption (public key/private key). |
| **culture; cultural practices** | **culture:** A human institution manifested in the learned behavior of people, including their specific belief systems, language(s), social relations, technologies, institutions, organizations, and systems for using and developing resources. [Source: National Council for the Social Studies, 2013]  **cultural practices**: The displays and behaviors of a culture. |
| **cybersecurity** | The protection against access to, or alteration of, computing resources through the use of technology, processes, and training. [Source: TechTarget] |
| **data** | The raw representation of variables. Data can be collected and used for reference or analysis. Data can be digital or non-digital and can be in many forms, including numbers, text, show of hands, images, sounds, or video. [Sources: Computing At School, 2013; Tech Terms] |
| **data structure** | A format for storing and accessing data within a computer program for the appropriate access and modification of the data. |
| **data type** | A classification of data that is distinguished by its attributes and the types of operations that can be performed on it. Some common data types are integer, string, Boolean (*true* or *false*), and floating-point. |
| **debugging** | The process of finding and correcting errors (bugs) in programs. [Source: Massachusetts Department of Elementary and Secondary Education, 2016]  *See also* bug. |
| **declare a variable** | In computer programming, declaring a variable determines the name and, in some programming languages, its data type. Programmers declare variables by writing the name of the variable into code. [Source: Techopedia]  *See also* variable. |
| **decompose; decomposition** | **decompose** *(v)*: To break down into components.  **decomposition** *(n)*: The act of breaking down a problem or system into components. [Source: Massachusetts Department of Elementary and Secondary Education, 2016] |
| **denial-of-service (DoS) attack** | A cyber-attack where the perpetrator seeks to make a computer system unavailable to its intended users. Denial of service is typically accomplished by flooding the computer system with excessive and/or invalid requests in an attempt to overload the system and prevent some or all legitimate requests from being fulfilled. [Sources: Techopedia; Tech Terms; Wikipedia] |
| **design** | Design is the creation of a plan or convention for the construction of an object, system or measurable human interaction (as in pseudocode and prototypes). [Source: Wikipedia] |
| **device** | A unit of physical hardware that provides one or more computing functions within a computing system. It can provide input to the computer, accept output, or both. [Source: Techopedia] |
| **digital** | A characteristic of electronic technology that uses binary digits 0 and 1, to generate, store, and process data. |
| **digital citizenship** | The norms of appropriate, responsible behavior with regard to the use of technology. [Source: Massachusetts Department of Elementary and Secondary Education, 2016] |
| **efficiency** | A measure of the amount of resources an algorithm uses to find an answer. It is usually expressed in terms of the theoretical computations, the memory used, the number of messages passed, the number of disk accesses, etc. [Source: National Institute of Standards and Technology – Dictionary of Algorithms and Data Structures] |
| **encapsulation** | The technique of combining data and the procedures that act on it to create a type.[Source: Free On-Line Dictionary of Computing]  *See also* structure. |
| **encoding** | The process of converting data into a format required for information processing. In computer technology, encoding is the process of applying a specific code, such as letters, symbols, and/or numbers, to data for conversion. [Source: Techopedia] |
| **encryption** | The conversion of electronic data into another form, called ciphertext, which cannot be easily understood by anyone except authorized parties. [Source: TechTarget] |
| **end user (or user)** | A person for whom a hardware or software product is designed (as distinguished from the developers). [Source: TechTarget]  *See also* audience. |
| **evaluate** | 1. Converting a programming statement into a value.  2. The process of examining the extent to which a computational artifact meets specified properties and goals. [Source: Free On-Line Dictionary of Computing] |
| **event** | Any identifiable occurrence that has significance for system hardware or software. User-generated events include keystrokes and mouse clicks; system-generated events include program loading and errors. [Source: TechTarget] |
| **event handler** | A procedure that specifies what should happen when a specific event occurs. |
| **execute; execution** | **execute:** To carry out (or “run*”*) an instruction or set of instructions (program, app, etc.).  **execution:** The process of executing an instruction or set of instructions.[Source: Free On-Line Dictionary of Computing] |
| **firewall** | A network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. A firewall typically establishes a barrier between a trusted internal network and an untrusted outside network, such as the Internet. Firewalls can be implemented as hardware, software, or a combination of both. [Sources: Webopedia; Wikipedia] |
| **floating point number** | The computing term used for representing non-integer numbers (e.g., numbers with decimal points). |
| **function** | *See definition for* procedure. |
| **garbage collection** | In computer science, garbage collection (GC) is a form of automatic memory management. The garbage collector, or just collector, attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program. [Source: Wikipedia] |
| **hardware** | The physical components that make up a computing system, computer, or computing device. [Source: Massachusetts Department of Elementary and Secondary Education, 2016] |
| **hierarchy** | An organizational structure in which items are ranked according to levels of importance. [Source: TechTarget] |
| **human–computer interaction (HCI)** | The study of how people interact with computers and to what extent computing systems are or are not developed for successful interaction with human beings. [Source: TechTarget] |
| **identifier** | The programmer-defined, unique name of a program element (such as a variable or procedure). An identifier name should indicate the meaning and usage of the element being named. [Source: Techopedia] |
| **implementation** | The process of expressing the design of a solution in a programming language (code) that can be made to run on a computing device. |
| **input** | Data sent to a computer program. |
| **integrated development environment (IDE)** | A software system for supporting the process of writing software. Such a system may include a syntax-directed editor, graphical tools for program entry, and integrated support for compiling and running the program and relating compilation errors back to the source. [Source: Free On-Line Dictionary of Computing] |
| **integrity** | The overall completeness, accuracy, and consistency of data. [Source: Techopedia] |
| **Internet** | The global collection of computer networks and their connections, all using shared protocols to communicate. [Source: Computing At School, 2013] |
| **Internet Protocol (IP) address** | A unique numerical label that is used to identify and locate each device connected to the Internet. [Tech Terms; Wikipedia] |
| **iterative** | Involving the repeating of a process with the aim of approaching a desired goal, target, or result. [Source: Massachusetts Department of Elementary and Secondary Education, 2016] |
| **library** | A collection of resources used by computer programs. These resources can include message templates, pre-written code, and specifications. The distinguishing feature is that a library is organized for the purposes of being reused by independent programs or sub-programs. [Source: Wikipedia] |
| **license** | An official permission or permit to do, use, or own something (as well as the document of that permission or permit). A public license is a license by which a copyright holder can grant additional copyright permissions to any and all persons in the general public as licensees. By applying a public license to a work, copyright holders give permission for others to copy or change their work in ways that would otherwise infringe copyright law. [Source: Wikipedia]  *See also* Creative Commons license. |
| **list** | A data structure holding many values, possibly of different types, which is usually accessed sequentially, working from the head to the end of the tail - an "ordered list". [Source: Free On-Line Dictionary of Computing] |
| **loop** | A programming structure that repeats a sequence of instructions as long as a specific condition is true. [Source: Tech Terms] |
| **Media Access Control (MAC) address** | A hardware identification number that uniquely identifies each device on a network. The MAC address is manufactured into every network card, such as an Ethernet card or a Wi-Fi card, and therefore cannot be changed. [Source: Tech Terms] |
| **memory** | Internal storage hardware used by computers to store and access data. |
| **model** | A representation (abstraction) of some part of a problem or a system. [Source: Massachusetts Department of Elementary and Secondary Education, 2016]  *Note: This definition differs from that used in science.* |
| **modularity** | The characteristic of a software/web application that has been divided *(decomposed)* into smaller modules. An application might have several procedures that are called from inside its main procedure. Existing procedures could be reused by recombining them in a new application. [Source: Techopedia] |
| **module** | Any of a number of distinct but interrelated units from which a program may be built up or into which a complex activity may be analyzed. A software component or part of a program that contains one or more procedures. One or more independently developed modules make up a program. [Source: Techopedia] |
| **nest(ed)** | To embed one object in another object. Nesting is quite common in programming, where different logic structures sequence, selection and loop) are combined (i.e., nested in one another). For example, nested loops are loops placed within loops, and nested conditionals allow the result of one conditional to lead to another. [Source: Webopedia] |
| **network** | A group of computing devices (personal computers, phones, servers, switches, routers, etc.) connected by cables or wireless media for the exchange of information and resources. |
| **object;**  **object-oriented programming (OOP)** | Object-oriented programming (OOP) refers to a type of computer programming (software design) in which programmers define data types that encapsulate both data and operations (procedures) that are applied to the data structure. In this way, the data structure becomes an object that includes both data and procedures. Programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects. [Source: Webopedia] |
| **operation** | An action, resulting from a single instruction, that changes the state of data. [Source: Free Dictionary] |
| **packet** | The unit of data sent over a network. [Source:Tech Terms] |
| **pair programming** | Pair programming is a software development technique in which two programmers work together using a single computer. One programmer acts as the driver by entering code, while the second programmer acts as the navigator, providing insight and feedback on the code as it is entered. The programmers switch roles on a regular basis.  *See also* program; programming. |
| **parameter** | A special kind of variable used in a procedure to refer to one of the pieces of data received as input by the procedure. [Source: Massachusetts Department of Elementary and Secondary Education, 2016] |
| **phishing** | The attempt to obtain sensitive information, such as usernames, passwords, and credit card details, by disguising as a trustworthy entity in an electronic communication. Phishing is typically carried out by email or instant messaging, and often directs users to enter personal information at a fake website which looks identical to the legitimate one except for the URL. [Source: Wikipedia] |
| **physical security tokens** | Devices used to gain access to an electronically restricted resource. The token is used in addition to, or in place of, a password. It acts like an electronic key to access something. An example of a physical security token is a wireless keycard opening a locked door. [Source: Wikipedia] |
| **piracy** | The illegal copying, distribution, or use of software. [Source: TechTarget] |
| **procedure** | An independent code module that fulfills some concrete task and is referenced within a larger body of program code. The fundamental role of a procedure is to offer a single point of reference for some small goal or task that the developer or programmer can trigger by invoking the procedure itself. [Source: Techopedia]  Note: In these standards, *procedure* is used as a general term that may refer to an actual procedure or a method, function, or module of any other name by which modules are known in other programming languages. |
| **process** | A series of actions or steps taken to achieve a particular outcome. [Source: Oxford] |
| **program; programming** | **program** *(n)*: A set of instructions that the computer executes to achieve a particular objective. [Source: Massachusetts Department of Elementary and Secondary Education, 2016]  **program** *(v)*: To produce a program by programming.  **programming**: The craft of analyzing problems and designing, writing, testing, and maintaining programs to solve them. [Source: Massachusetts Department of Elementary and Secondary Education, 2016] |
| **protocol** | The special set of rules used by endpoints in a telecommunication connection when they communicate. Protocols specify interactions between the communicating entities. [Source: TechTarget] |
| **prototype** | An early approximation of a final product or information system, often built for demonstration purposes. [Sources: TechTarget, Techopedia] |
| **pseudocode** | An informal high-level description of a computer program or other algorithm. Pseudocode does not require strict syntax and uses natural language. The purpose of using pseudocode is that it is easier for people to understand than conventional programming language code. It is commonly used in computer program development for sketching out the structure of the program before actual coding takes place. [Sources: Tech Terms; Wikipedia] |
| **public key encryption** | An encryption technique that uses pairs of keys: *public keys*, which may be widely known, and *private keys*, which are known only to the owner. In a public key encryption system, any sender can encrypt a message using the public key of the receiver, but the message can only be decrypted with the receiver’s private key. [Sources: Tech Terms; Wikipedia] |
| **ransomware** | A type of malicious software that blocks access to a victim’s computer or certain files unless a ransom is paid. [Sources: Tech Terms; Wikipedia] |
| **recursion; recursive procedures** | A powerful problem solving approach where the problem solution is built on solutions of smaller instances of the same problem. A base case, which returns a result without referencing itself, must be defined, otherwise infinite recursion will occur.  Procedures that incorporate recursion are called recursive procedures.  *See also* procedure. |
| **redundancy** | A system design in which a component is duplicated, so if it fails, there will be a backup. [Source: TechTarget] |
| **reliability** | An attribute of any system that consistently produces the same results given the same configuration, preferably meeting or exceeding its requirements. [Source: Free On-Line Dictionary of Computing] |
| **remix** | The process of creating something new from something old. Originally a process that involved music, remixing involves creating a new version of a program by recombining and modifying parts of existing programs, and often adding new pieces, to form new solutions. [Source: Kafai & Burke, 2014] |
| **router** | A device or software that determines the path that data packets travel from source to destination. [Source: TechTarget] |
| **scalability** | The capability of a network or a program to handle a growing amount of work or its potential to be enlarged to accommodate that growth. [Source: Wikipedia] |
| **scrape a web page** | Scraping a web page is the process of automatically mining data or collecting information from the World Wide Web. Current web scraping solutions range from the ad-hoc, requiring human effort, to fully automated systems that are able to convert entire web sites into structured information, with limitations. [Source: Wikipedia] |
| **security** | *See the definition for* cybersecurity. |
| **server** | A computer program or device that provides services to other computer programs or devices. [Sources: TechTarget; Wikipedia] |
| **set** | A data type that can store a collection of values, without any particular order, and no repeated values. [Source: Wikipedia] |
| **simulate; simulation** | **simulate** *(v)***:** To imitate the operation of a real-world process or system.  **simulation** *(n)*: Imitation of the operation of a real-world process or system. [Source: Massachusetts Department of Elementary and Secondary Education, 2016] |
| **social engineering** | A non-technical method of breaking into a secured computer system. Victims of social engineering are tricked into releasing information that they do not realize will be used to attack a computer network. Phishing is a type of security attack that relies on social engineering. [Sources: Techopedia; Webopedia]  *See also* phishing. |
| **software** | Programs that run on a computing system, computer, or other computing device. |
| **spyware** | Software that aims to gather information about a person or organization without their knowledge, that may send such information to another entity without the consumer’s consent, or that asserts control over a device without the consumer’s knowledge. [Source: Wikipedia] |
| **storage** | *(place)* A place, usually a device, into which data can be entered, in which the data can be held, and from which the data can be retrieved at a later time. [Source: Free On-Line Dictionary of Computing]  *(process)* A process through which digital data is saved within a data storage device by means of computing technology. Storage is a mechanism that enables a computer to retain data, either temporarily or permanently. [Source: Techopedia] |
| **storyboard** | A storyboard is a graphic organizer in the form of illustrations or images displayed in sequence for the purpose of pre-visualizing a motion picture, animation, motion graphic, interactive media sequence, or other computational artifact. [Source: Wikipedia] |
| **string** | A sequence of letters, numbers, and/or other symbols. A string might represent, for example, a name, address, or song title. Some procedures commonly associated with strings are length, concatenation, and substring. [Source: TechTarget] |
| **structure** | A general term used in the framework to discuss the concept of encapsulation without specifying a particular programming methodology.  *See also* encapsulation. |
| **subroutine** | *See definition for* procedure. |
| **switch** | A high-speed device that receives incoming data packets and redirects them to their destination on a local area network (LAN). [Source: Techopedia] |
| **system** | A collection of elements or components that work together for a common purpose. [Source: TechTarget]  *See also* computing system. |
| **system** | A collection of elements or components that work together for a common purpose, [Source: Tech Target]  *See also* computing system. |
| **test case** | A set of conditions or variables under which a tester will determine whether the system being tested satisfies requirements or works correctly. [Source: Software Testing Fundamentals] |
| **testing** | The process of verifying that a software program works as expected. [Source: Wikipedia] |
| **topology** | The physical and logical configuration of a network; the arrangement of a network, including its nodes and connecting links. A logical topology is the way devices appear connected to the user. A physical topology is the way they are actually interconnected with wires and cables. [Source: PCMag] |
| **transmission** | Data transmission is the process of sending digital or analog data over a communication medium to one or more computing, network, communication or electronic devices. [Source: Techopedia] |
| **troubleshooting** | A systematic approach to problem solving that is often used to find and resolve a problem, error, or fault within software or a computing system. [Sources: Techopedia, TechTarget] |
| **two-factor authentication** | A security system that requires more than one method of authentication from independent categories of credentials to verify the user’s identity for a login or other transaction. Multi-factor authentication (which includes two-factor authentication) combines two or more independent credentials: what the user knows (password), what the user has (physical security token), and what the user is (biometric verification). [Source: TechTarget] |
| **unplugged** | An approach to teaching computer science without computers where concepts are presented through engaging activities and puzzles by using cards, crayons, and active playing. [Source: Hazzan, Lapidot, & Ragonis, 2015] |
| **user** | *See the definition for* end user. |
| **variable** | A memory location that stores a value. A variable is associated with a symbolic name (or identifier) and a data type. Variables are not just used for numbers; they can also hold text, including whole sentences *(strings*) or logical values (*true* or *false*). The value of a variable is normally changed during the course of program execution. [Sources: Computing At School, 2013; Reges & Stepp, 2014; Techopedia; Wikipedia]  *Note: This definition differs from that used in math and statistics.*  *See also* declare a variable. |
| **version control software** | Version control is used to manage multiple versions of computer files and programs. Version control software provides two primary data management capabilities. It allows users to (a) lock files so they can only be edited by one person at a time, and (b) track changes to files. [Source: TechTerms] |
| **virus** | A type of malicious software program that, when executed, replicates itself by modifying other computer programs and inserting its own code. When this replication succeeds, the affected computers are then said to be “infected” with a computer virus. [Source: Wikipedia] |
| **web filter** | A program that can screen an incoming web page to determine whether some or all of it should not be displayed to the user. The filter checks the origin or content of a web page against a set of rules provided by the company or person who installed the web filter. A web filter allows an enterprise or individual to block out pages from web sites that are likely to include objectionable content. [Source: TechTarget] |
| **worm** | A type of computer virus that replicates itself to spread to uninfected computers, but does not alter any files on the computer. However, worms can still cause havoc by multiplying so many times that it takes up all the computer’s available memory or hard disk space. [Sources: TechTarget, Tech Terms] |

### References

*Some definitions came directly from these sources, while others were excerpted or adapted to include content relevant to this framework.*

Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). *A model curriculum for K–12 computer science: Report of the ACM K–12 task force curriculum committee* (2nd ed.). New York, NY: Association for Computing Machinery.

The Joint Task Force for Computing Curricula 2005. (2006). *Computing curricula 2005: The overview report covering undergraduate degree programs in computer engineering, computer science, information systems, information technology, software engineering.* Retrieved from <https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2005-march06final.pdf>.

Computing At School. (2013). *Computing in the national curriculum: A guide for primary teachers.* Belford, UK: Newnorth Print. <http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf>.

College Board. (2016). *AP Computer Science Principles course and exam description.* New York, NY: College Board.<https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf>.

Free on-line dictionary of computing. (n.d.). Retrieved from [http://foldoc.org](http://foldoc.org/).

The free dictionary. (n.d.) Retrieved from <http://www.thefreedictionary.com/operation>.

Hazzan, O., Lapidot, T., & Ragonis, N. (2015). Guide to Teaching Computer Science: An Activity-Based Approach (2nd ed. 2014 edition). London: Springer.

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

Kafai, Y., & Burke, Q. (2014). Connected code: Why children need to learn programming. Cambridge, MA: MIT Press.

Lee, I. (2016). Reclaiming the roots of CT. CSTA Voice: The Voice of K–12 Computer Science Education and Its Educators, 12(1), 3–4. ~~http://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/Voice/csta\_voice\_03\_2016.pdf~~. [no longer available]

Massachusetts Department of Elementary and Secondary Education. (2016, June). 2016 Massachusetts digital literacy and computer science (DLCS) curriculum framework. Malden, MA: Author. <http://www.doe.mass.edu/frameworks/dlcs.pdf>.

National Council for the Social Studies. (2013). The college, career, and civic life (C3) framework for social studies state standards: Guidance for enhancing the rigor of K–12 civics, economics*, geography, and history.* Silver Spring, MD: Author. <http://www.socialstudies.org/system/files/c3/C3-Framework-for-Social-Studies.pdf>.

Computational Modeling. (2016). Retrieved from <https://www.nibib.nih.gov/science-education/science-topics/computational-modeling>.

Pieterse, V., & Black, P. E. (Eds.). (n.d). *Dictionary of algorithms and data structures.* Retrieved from [https://xlinux.nist.gov/dads//](https://xlinux.nist.gov/dads/).

Oxford dictionaries. (n.d.). Retrieved from <http://www.oxforddictionaries.com/us>.

PCmag.com encyclopedia. (n.d.). Retrieved from <http://www.pcmag.com/encyclopedia/term/46301/logical-vs-physical-topology>.

Reges, S. & Stepp, M. (2014). *Building Java programs: A back to basics approach.* Pearson Education.

Ross, B. (2016, May 10). What is automation and how can it improve customer service? *Information Age.* Retrieved from <http://www.information-age.com/industry/software/123461408/what-automation-and-how-can-it-improve-customer-service>.

Software testing fundamentals. (n.d). Retrieved from [http://softwaretestingfundamentals.com](http://softwaretestingfundamentals.com/).

Tech terms computer dictionary. (n.d.). Retrieved from [http://www.techterms.com](http://www.techterms.com/).

[Techopedia technology dictionary. (n.d.). Retrieved from https://www.techopedia.com/dictionary](https://www.techopedia.com/dictionary).

TechTarget network. (n.d.)[. Retrieved from http://www.techtarget.com/network](http://www.techtarget.com/network).

Webopedia. (n.d.). Retrieved from [http://www.webopedia.com](http://www.webopedia.com/).

Wikipedia: The free encyclopedia. (n.d.). Retrieved from <https://www.wikipedia.org/>.

California Department of Education: August 2018