[INTERIM] CSTA K–12 COMPUTER SCIENCE STANDARDS

REVISED 2016

CSTA STANDARDS TASK FORCE

CSTEACHERS.ORG
COMPUTER SCIENCE TEACHERS ASSOCIATION

Association for Computing Machinery

Computer Science Teachers Association (CSTA)
The Association for Computing Machinery, Inc. (ACM)
2 Penn Plaza, Suite 701
New York, NY 10121

CSTA/ACM ISBN: # 978-1-4503-4762-4
CSTA/ACM Order Number: #999160
Cost: $7.00 US/$8.00 International

Additional copies may be ordered prepaid from:

CSTA Order Department
Attn: CSTA Standards
PO Box 767, Springfield, OR 97477
Email: customerservice@csta-hq.org
Phone: +1-800-342-6626 | Fax: +1-541-944-1318

2015–2016 CSTA STANDARDS REVISION TASK FORCE

Leadership

Deborah Seehorn
Co-Chair, CSTA Board of Directors Past Chair, Retired
North Carolina Department of Public Instruction

Tammy Pirmann
Co-Chair, CSTA Board of Directors School District Representative, Springfield Township, PA

K–5 Team

Todd Lash
Grade Level Lead, K–5
Instructional Coach and Teaching Specialist for Computer Science and Computational Thinking
Kenwood Elementary, IL

Leticia Batista
Kindergarten Teacher
Apple Distinguished Educator
McKinna Elementary, Oxnard, CA

Dylan Ryder
Intermediate Division
Educational Technologist
The School at Columbia University

Vicky Sedgwick
K–8 Technology Teacher and Tech Trainer
St. Martin's Episcopal School, CA

6–8 Team

Irene Lee
Grade Level Lead, 6–8
Research Scientist
MIT's Scheller Teacher Education Program/Education Arcade, MA

Dianne O'Grady-Cunniff
Instructional Specialist for Computer Science and Technology Education
Charles County Public Schools, MD

Bryan Twarek
Computer Science Coordinator
San Francisco Unified School District

9–12 Team

Daniel Moix
Grade Level Lead, 9–12
Arkansas K–12 Computer Science Education Specialist
Arkansas School for Mathematics, Sciences & Arts

Julia Bell
Associate Professor of Computer Science
Walters State Community College, TN

Laura Blankenship
Chair of Computer Science and Interim Dean of Academic Affairs
The Baldwin School, PA

Lori Pollock
Professor of Computer and Information Sciences
University of Delaware

Chinma Uche
Computer Science and Math Teacher
Greater Hartford Academy of Mathematics and Science and CREC's Academy of Aerospace and Engineering, CT

ACKNOWLEDGEMENTS

The 2015–2016 CSTA K–12 CS Standards Review Task Force Co-Chairs gratefully acknowledge the pioneers in K–12 computer science education who contributed to the revision of the CSTA K–12 CS Standards. The groups listed below graciously shared their time and expertise with the task force members, thereby providing inspiration and guidance as the revisions took place. We salute you for your dedication to K–12 computer science education and appreciate your collaborative spirit.

- 2015–2016 CSTA K–12 CS Standards Revision Task Force Members
- CSTA Membership Community
- Achieve
- Code.org
- K–12 CS Framework
- Florida Department of Education
- Maryland CS Matters Steering Committee
- Washington Department of Education and University of Washington CS Faculty
- Anthony Owen, Arkansas Department of Education
- Jim Stanton, MassCAN
- Volunteer Reviewers, including
  - Chris Stephenson
  - Deepa Muralidhar
  - Sheena Vaidyanathan
  - Debbie Carter

We also wish to extend our heartfelt gratitude to Google, one of CSTA's most supportive partners. Thank you for providing us with the capacity to work on this important project. We are grateful for your contributions and support.

DEDICATION

The 2015–2016 CSTA Standards Revision Task Force dedicates this revision of the CSTA K–12 Computer Science Standards to our friend and colleague, Karen Marie Putman. Karen was an ardent supporter of CSTA and the CSTA K–12 CS standards. She was a Computer Science teacher at the University of Chicago Laboratory School, where she had taught for 45 years. She was one of the earliest K–8 Computer Science educators, transitioning from teaching German to teaching Logo Programming.

Karen enthusiastically joined the 2015–2016 CSTA Standards Revision Task Force in early September 2015. She was serving as the K–5 Grade Level Lead on the task force at the time of her sudden and untimely death. We have lost a special CS educator who had much history and knowledge in the field of K–8 CS education. The task force members wish to dedicate these standards in memory of Karen, who would have loved to see the standards come to fruition.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Although the recent flurry of activity and exposure around computer science (CS) education has increased visibility in recent years, the general public, administration, and most legislators are not as well educated about CS as they should be, and a serious shortage of information about computer science exists at all levels and may continue into the foreseeable future. The [Interim] CSTA K–12 Computer Science Standards aim to help address these problems. They provide a guide within which state departments of education and school districts can revise their curricula to better address the need to educate young people in this important subject area.

These interim standards provide a three-level guide for computer science. The first two levels are aimed at grades K–5 and 6–8, respectively. It is expected that the learning outcomes in Level 1 (K–5) will be addressed cross-curricularly, that is, in the context of other academic subjects. The learning outcomes in Level 2 (6–8) may be addressed either cross-curricularly or in discrete computer science courses. Level 3 is divided into two separate levels. Level 3A (9–10) represents what *all* students should know and be able to do by the time they graduate high school. Level 3B (11–12) represents the progression of learning for students who express an interest in further study of computer science.

We believe that computational thinking is a problem solving methodology that expands the realm of computer science into all disciplines, providing a distinct means of analyzing and developing solutions to problems that can be solved computationally. With its focus on abstraction, automation, and analysis, **computational thinking is a core element of the broader discipline of computer science** and for that reason it is interwoven through these computer science standards at all levels of K–12 learning.

These recommendations are not made in a vacuum. We understand the serious constraints under which school districts are operating and the uphill battle that computer science faces in the light of other priorities, as well as time and budget constraints. Many follow-up efforts are still needed, however, to sustain the momentum we hope these standards will generate. Teacher professional development, curriculum innovation, in-class testing, textbook and website development, and dissemination are but a few of the challenges.

We hope these standards will serve as a catalyst for widespread discussions and the initiation of many projects that can take the evolution of K–12 computer science to the next level. We invite you to read the entire document, and then to take part in this discussion in a way that mutually benefits both you and the K–12 education community. More information about ongoing activities to support computer science education in K–12 can be found at http://www.csteachers.org

INTRODUCTION

Purpose & Objectives

The purpose of this document is to set forth, at all stages of their learning, the knowledge and skills that students must have to enable them to thrive in this new global information economy. It defines a set of national standards for K–12 computer science and suggests steps that will be needed to enable their wide implementation. It is intended to introduce the principles and methodologies of computer science to all students, whether they are college bound or career bound after high school. The interim standards outlined in this document address the entire K–12 range. They complement existing K–12 computer science and IT curricula where they are already established, especially the advanced placement (AP) computer science curricula (AP, 2010). Additionally, the standards complement existing curricula in other disciplines.

The Interim K–12 CS Standards are intended to:

1. introduce the fundamental concepts of computer science to all students, beginning at the elementary school level;
2. present computer science at the secondary school level in a way that will be both accessible and worthy of a computer science credit, or, for the Level 3B courses, as a required graduation credit for math or science;
3. offer additional secondary-level computer science standards that will allow interested students to study facets of computer science in depth and prepare them for entry into a career or college;
4. increase the knowledge of computer science for all students, especially those who are members of underrepresented groups.

Revision Process

All drafts of this report have been informed by feedback from many sources; we hope that this interim draft will receive widespread dissemination and continued scrutiny from everyone who has interests or experience in K–12 computer science education. To that end, these standards are published online, along with feedback that has been actively sought from many professional organizations, curriculum experts, and community members. For more information on the process, please visit the CSTA website at http://www.csteachers.org/CSTA_Standards.

We recognize that many of the recommendations in this report are ambitious but we believe that they are critical to ensuring that students achieve the necessary level of knowledge, skills, and experience. We offer this work as a comprehensive and coherent set of standards—an ideal toward which many districts can evolve over time. This report thus provides a catalyst for a long-term process. It defines the "what" from which the "how" can follow during the next several years.

Implementation Challenges

Teaching any subject effectively depends on the existence of sound learning standards for students, explicit teacher certification standards, appropriate teacher professional development programs, and effective curricular materials. K–12 computer science education faces unique challenges along these lines.

For schools to widely implement these standards, work is needed in three important areas: teacher preparation, state-level content standards, and curriculum materials development. In addition, persons in local and state leadership positions must acknowledge the importance of computer science education for the future of our society. States and accrediting organizations should make this a factor in overall school accreditation. Some states have begun to establish computer science content standards, define models for teacher certification, provide in-service professional development in computer science, and experiment with developing new curricular materials. However, a much wider and continued effort and commitment are required.

Improving computer science education is a significant challenge that will require attention and interventions from multiple institutions. Professional organizations in computer science can make an important contribution. CSTA, for example, is a professional organization that supports and promotes the teaching of computer science and other computing disciplines. CSTA provides a large number of programs that include the development and dissemination of learning resources, the provision of professional development, and advocating for state and federal level policies to improve computer science education. Other organizations, such as ACM, the IEEE Computer Society, institutions of higher education, and national and local teacher organizations, can also work toward addressing these issues in K–12 computer science education.

Industry is also deeply affected by pipeline issues and the need to produce workers who have the skills needed to support and build the technology tools of the future. It is therefore in their best interest to contribute significantly to improving access and quality of computer science courses in K–12.

Conclusion

Computer science education is a dynamic discipline. This document serves as an *interim* draft document and represents changes to the previous standards in order to address recent changes in the computer science education landscape. The launch of the K–12 CS Framework, the Maker movement, and a national focus on cybersecurity are just some of the changes that are still in motion during this document's release.

Due to rapid growth and changes in our field, computer science standards cannot be static. These standards must be reviewed and updated on a regular basis, and not considered complete and finalized. CSTA is committed to an inclusive, iterative process that allows multiple drafts and revisions of the CSTA K–12 CS Standards. CSTA is one of several supporters of the K–12 CS Framework project. The Standards influence the Framework and the Framework influences the Standards. For more information, visit the CSTA Standards FAQ at www.csteachers.org/2016standards.

It is not an exaggeration to say that our lives depend upon computer systems and the people who maintain them to keep us safe on the road and in air, help physicians diagnose and treat health care problems, and play a critical role in the design of new drug therapies. A fundamental understanding of computer science enables students to be both educated users of technology and innovators capable of designing computing systems to improve the quality of life for everyone.

We understand that many obstacles lie in the way of the ideal of a K–12 computer science education for all students. How will room be found in the jam-packed curriculum? How will qualified teachers be recruited, trained, and credentialed? In the world of standards-centric evaluation of schools, should computer science support existing standards, or should new ones be designed for computer science? These and other questions and challenges are significant, but so are the benefits—to students and to society—of computer science finding its rightful place as a component of high-quality education for all students.

INTERIM 2016 CSTA K–12 CS Standards
Level 1 (Grades K–5)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
K–2	1A-A-7-1	Give credit when using code, music, or pictures (for example) that were created by others.	Algorithms and Programs	Communicating about Computing
K–2	1A-A-5-2	Construct programs, to accomplish a task or as a means of creative expression, which include sequencing, events, and simple loops, using a block-based visual programming language, both independently and collaboratively (e.g., pair programming).	Algorithms and Programs	Creating Computational Artifacts
K–2	1A-A-5-3	Plan and create a design document to illustrate thoughts, ideas, and stories in a sequential (step-by-step) manner (e.g., story map, storyboard, sequential graphic organizer).	Algorithms and Programs	Creating Computational Artifacts
K–2	1A-A-4-4	Use numbers or other symbols to represent data (e.g., thumbs up/down for yes/no, color by number, arrows for direction, encoding/decoding a word using numbers or pictographs).	Algorithms and Programs	Developing and Using Abstractions
K–2	1A-A-3-5	Decompose (break down) a larger problem into smaller sub-problems with teacher guidance or independently.	Algorithms and Programs	Recognizing and Defining Computational Problems
K–2	1A-A-3-6	Categorize a group of items based on the attributes or actions of each item, with or without a computing device.	Algorithms and Programs	Recognizing and Defining Computational Problems

INTERIM 2016 CSTA K–12 CS Standards
Level 1 (Grades K–5)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
K–2	1A-A-3-7	Construct and execute algorithms (sets of step-by-step instructions) that include sequencing and simple loops to accomplish a task, both independently and collaboratively, with or without a computing device.	Algorithms and Programs	Recognizing and Defining Computational Problems
K–2	1A-A-6-8	Analyze and debug (fix) an algorithm that includes sequencing and simple loops, with or without a computing device.	Algorithms and Programs	Testing and Refining
K–2	1A-C-7-9	Identify and use software that controls computational devices (e.g., use an app to draw on the screen, use software to write a story or control robots).	Computing Systems	Communicating about Computing
K–2	1A-C-7-10	Use appropriate terminology in naming and describing the function of common computing devices and components (e.g., desktop computer, laptop computer, tablet device, monitor, keyboard, mouse, printer).	Computing Systems	Communicating about Computing
K–2	1A-C-6-11	Identify, using accurate terminology, simple hardware and software problems that may occur during use (e.g., app or program not working as expected, no sound, device won't turn on).	Computing Systems	Testing and Refining
K–2	1A-D-7-12	Collect data over time and organize it in a chart or graph in order to make a prediction.	Data and Analysis	Communicating About Computing
K–2	1A-D-4-13	Use a computing device to store, search, retrieve, modify, and delete information and define the information stored as data.	Data and Analysis	Developing and Using Abstractions
K–2	1A-D-4-14	Create a model of an object or process in order to identify patterns and essential elements (e.g., water cycle, butterfly life cycle, seasonal weather patterns).	Data and Analysis	Developing and Using Abstractions

INTERIM 2016 CSTA K–12 CS Standards
Level 1 (Grades K–5)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
K–2	1A-I-7-15	Compare and contrast examples of how computing technology has changed and improved the way people live, work, and interact.	Impacts of Computing	Communicating about Computing
K–2	1A-N-2-16	Use computers or other computing devices to connect with people using a network (e.g., the Internet) to communicate, access, and share information as a class.	Networks and the Internet	Collaborating
K–2	1A-N-7-17	Use passwords to protect private information and discuss the effects of password misuse.	Networks and the Internet	Communicating about Computing

INTERIM 2016 CSTA K–12 CS Standards
Level 1 (Grades K–5)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
3–5	1B-A-2-1	Apply collaboration strategies to support problem solving within the design cycle of a program.	Algorithms and Programs	Collaborating
3–5	1B-A-7-2	Use proper citations and document when ideas are borrowed and changed for their own use (e.g., using pictures created by others, using music created by others, remixing programming projects).	Algorithms and Programs	Communicating about Computing
3–5	1B-A-5-3	Create a plan as part of the iterative design process, both independently and with diverse collaborative teams (e.g., storyboard, flowchart, pseudo-code, story map).	Algorithms and Programs	Creating Computational Artifacts
3–5	1B-A-5-4	Construct programs, in order to solve a problem or for creative expression, that include sequencing, events, loops, conditionals, parallelism, and variables, using a block-based visual programming language or text-based language, both independently and collaboratively (e.g., pair programming).	Algorithms and Programs	Creating Computational Artifacts
3–5	1B-A-5-5	Use mathematical operations to change a value stored in a variable.	Algorithms and Programs	Creating Computational Artifacts
3–5	1B-A-3-6	Decompose (break down) a larger problem into smaller sub-problems, independently or in a collaborative group.	Algorithms and Programs	Recognizing and Defining Computational Problems

INTERIM 2016 CSTA K–12 CS Standards
Level 1 (Grades K–5)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
3–5	1B-A-3-7	Construct and execute an algorithm (set of step-by-step instructions) that includes sequencing, loops, and conditionals to accomplish a task, both independently and collaboratively, with or without a computing device.	Algorithms and Programs	Recognizing and Defining Computational Problems
3–5	1B-A-6-8	Analyze and debug (fix) an algorithm that includes sequencing, events, loops, conditionals, parallelism, and variables.	Algorithms and Programs	Testing and Refining
3–5	1B-C-7-9	Model how a computer system works. [Clarification: Only includes basic elements of a computer system, such as input, output, processor, sensors, and storage.]	Computing Systems	Communicating about Computing
3–5	1B-C-7-10	Use appropriate terminology in naming internal and external components of computing devices and describing their relationships, capabilities, and limitations.	Computing Systems	Communicating about Computing
3–5	1B-C-6-11	Identify, using accurate terminology, simple hardware and software problems that may occur during use, and apply strategies for solving problems (e.g., reboot device, check for power, check network availability, close and reopen app).	Computing Systems	Testing and Refining
3–5	1B-D-5-12	Create a computational artifact to model the attributes and behaviors associated with a concept (e.g., solar system, life cycle of a plant).	Data and Analysis	Creating Computational Artifacts
3–5	1B-D-5-13	Answer a question by using a computer to manipulate (e.g., sort, total and/or average, chart, graph) and analyze data that has been collected by the class or student.	Data and Analysis	Creating Computational Artifacts

INTERIM 2016 CSTA K–12 CS Standards
Level 1 (Grades K–5)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA Standard	Framework Concept	Framework Practice
3–5	1B-D-4-14	Use numeric values to represent non-numeric ideas in the computer (binary, ASCII, pixel attributes such as RGB).	Data and Analysis	Developing and Using Abstractions
3–5	1B-I-7-15	Evaluate and describe the positive and negative impacts of the pervasiveness of computers and computing in daily life (e.g., downloading videos and audio files, electronic appliances, wireless Internet, mobile computing devices, GPS systems, wearable computing).	Impacts of Computing	Communicating about Computing
3–5	1B-I-7-16	Generate examples of how computing can affect society, and also how societal values can shape computing choices.	Impacts of Computing	Communicating about Computing
3–5	1B-I-1-17	Seek out and compare diverse perspectives, synchronously or asynchronously, to improve a project.	Impacts of Computing	Fostering an Inclusive Computing Culture
3–5	1B-I-1-18	Brainstorm ways in which computing devices could be made more accessible to all users.	Impacts of Computing	Fostering an Inclusive Computing Culture
3–5	1B-I-1-19	Explain problems that relate to using computing devices and networks (e.g., logging out to deter others from using your account, cyberbullying, privacy of personal information, and ownership).	Impacts of Computing	Fostering an Inclusive Computing Culture
3–5	1B-N-7-20	Create examples of strong passwords, explain why strong passwords should be used, and demonstrate proper use and protection of personal passwords.	Networks and the Internet	Communicating about Computing
3–5	1B-N-4-21	Model how a device on a network sends a message from one device (sender) to another (receiver) while following specific rules.	Networks and the Internet	Developing and Using Abstractions

INTERIM 2016 CSTA K–12 CS Standards
Level 2 (Grades 6–8)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
6–8	2-A-2-1	Solicit and integrate peer feedback as appropriate to develop or refine a program.	Algorithms and Programming	Collaborating
6–8	2-A-7-2	Compare different algorithms that may be used to solve the same problem in terms of their speed, clarity, and size (e.g., different algorithms solve the same problem, but one might be faster than the other). [Clarification: Students are not expected to quantify these differences.]	Algorithms and Programming	Communicating about Computing
6–8	2-A-7-3	Provide proper attribution when code is borrowed or built upon.	Algorithms and Programming	Communicating about Computing
6–8	2-A-7-4	Interpret the flow of execution of algorithms and predict their outcomes. [Clarification: Algorithms can be expressed using natural language, flow and control diagrams, comments within code, and pseudocode.]	Algorithms and Programming	Communicating about Computing
6–8	2-A-5-5	Design, develop, and present computational artifacts such as mobile applications that address social problems both independently and collaboratively.	Algorithms and Programming	Creating Computational Artifacts
6–8	2-A-5-6	Develop programs, both independently and collaboratively, that include sequences with nested loops and multiple branches. [Clarification: At this level, students may use block-based and/or text-based programming languages.]	Algorithms and Programming	Creating Computational Artifacts

INTERIM 2016 CSTA K–12 CS Standards
Level 2 (Grades 6–8)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
6–8	2-A-5-7	Create variables that represent different types of data and manipulate their values.	Algorithms and Programming	Creating Computational Artifacts
6–8	2-A-4-8	Define and use procedures that hide the complexity of a task and can be reused to solve similar tasks. [Clarification: Students use and modify, but do not necessarily create, procedures with parameters.]	Algorithms and Programming	Developing and Using Abstractions
6–8	2-A-3-9	Decompose a problem into parts and create solutions for each part.	Algorithms and Programming	Recognizing and Defining Computational Problems
6–8	2-A-6-10	Use an iterative design process (e.g., define the problem, generate ideas, build, test, and improve solutions) to solve problems, both independently and collaboratively.	Algorithms and Programming	Testing and Refining
6–8	2-C-7-11	Justify the hardware and software chosen to accomplish a task (e.g., comparison of the features of a tablet vs. desktop, selecting which sensors and platform to use in building a robot or developing a mobile app).	Computing Systems	Communicating about Computing
6–8	2-C-4-12	Analyze the relationship between a device's computational components and its capabilities. [Clarification: Computing Systems include not only computers, but also cars, microwaves, smartphones, traffic lights, and flash drives.]	Computing Systems	Developing and Using Abstractions

INTERIM 2016 CSTA K–12 CS Standards
Level 2 (Grades 6–8)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
6–8	2-C-6-13	Use a systematic process to identify the source of a problem within individual and connected devices (e.g., follow a troubleshooting flow diagram, make changes to software to see if hardware will work, restart device, check connections, swap in working components).	Computing Systems	Testing and Refining
6–8	2-D-7-14	Describe how different formats of stored data represent tradeoffs between quality and size. [Clarification: compare examples of music, text and/or image formats.]	Data and Analysis	Communicating about Computing
6–8	2-D-7-15	Explain the processes used to collect, transform, and analyze data to solve a problem using computational tools (e.g., use an app or spreadsheet form to collect data, decide which data to use or ignore, and choose a visualization method.).	Data and Analysis	Communicating about Computing
6–8	2-D-5-16	Revise computational models to more accurately reflect real-world systems (e.g., ecosystems, epidemics, spread of ideas).	Data and Analysis	Creating Computational Artifacts
6–8	2-D-4-17	Represent data using different encoding schemes (e.g., binary, Unicode, Morse code, shorthand, student-created codes).	Data and Analysis	Developing and Using Abstractions
6–8	2-I-7-18	Summarize negative and positive impacts of using data and information to categorize people, predict behavior, and make recommendations based on those predictions (e.g., customizing search results or targeted advertising, based on previous browsing history, can save search time and limit options at the same time).	Impacts of Computing	Communicating about Computing
6–8	2-I-7-19	Explain how computer science fosters innovation and enhances nearly all careers and disciplines.	Impacts of Computing	Communicating about Computing

INTERIM 2016 CSTA K–12 CS Standards
Level 2 (Grades 6–8)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
6–8	2-I-1-20	Provide examples of how computational artifacts and devices impact health and well-being, both positively and negatively.	Impacts of Computing	Fostering an Inclusive Computing Culture
6–8	2-I-1-21	Describe ways in which the Internet impacts global communication and collaborating.	Impacts of Computing	Fostering an Inclusive Computing Culture
6–8	2-I-1-22	Describe ethical issues that relate to computing devices and networks (e.g., equity of access, security, hacking, intellectual property, copyright, Creative Commons licensing, and plagiarism).	Impacts of Computing	Fostering an Inclusive Computing Culture
6–8	2-I-6-23	Redesign a computational artifact to remove barriers to universal access (e.g., using captions on images, high contrast colors, and/or larger font sizes).	Impacts of Computing	Testing and Refining
6–8	2-N-7-24	Summarize security risks associated with weak passwords, lack of encryption, insecure transactions, and persistence of data.	Networks and the Internet	Communicating about Computing
6–8	2-N-4-25	Simulate how information is transmitted as packets through multiple devices over the Internet and networks.	Networks and the Internet	Developing and Using Abstractions

INTERIM 2016 CSTA K–12 CS Standards
Level 3A (Grades 9–10)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
9–10	3A-A-2-1	Design and develop a software artifact working in a team.	Algorithms and Programming	Collaborating
9–10	3A-A-2-2	Demonstrate how diverse collaborating impacts the design and development of software products (e.g., discussing real-world examples of products that have been improved through having a diverse design team or reflecting on their own team's development experience).	Algorithms and Programming	Collaborating
9–10	3A-A-7-3	Compare and contrast various software licensing schemes (e.g., open source, freeware, commercial).	Algorithms and Programming	Communicating about Computing
9–10	3A-A-5-4	Design, develop, and implement a computing artifact that responds to an event (e.g., robot that responds to a sensor, mobile app that responds to a text message, sprite that responds to a broadcast).	Algorithms and Programming	Creating Computational Artifacts
9–10	3A-A-5-5	Use user-centered research and design techniques (e.g., surveys, interviews) to create software solutions.	Algorithms and Programming	Creating Computational Artifacts
9–10	3A-A-5-6	Integrate grade-level appropriate mathematical techniques, concepts, and processes in the creation of computing artifacts.	Algorithms and Programming	Creating Computational Artifacts

INTERIM 2016 CSTA K–12 CS Standards
Level 3A (Grades 9–10)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
9–10	3A-A-4-7	Understand the notion of hierarchy and abstraction in high-level languages, translation, instruction sets, and logic circuits.	Algorithms and Programming	Developing and Using Abstractions
9–10	3A-A-4-8	Deconstruct a complex problem into simpler parts using predefined constructs (e.g., functions and parameters and/or classes).	Algorithms and Programming	Developing and Using Abstractions
9–10	3A-A-4-9	Demonstrate the value of abstraction for managing problem complexity (e.g., using a list instead of discrete variables).	Algorithms and Programming	Developing and Using Abstractions
9–10	3A-A-3-10	Design algorithms using sequence, selection, and iteration.	Algorithms and Programming	Recognizing and Defining Computational Problems
9–10	3A-A-3-11	Explain and demonstrate how modeling and simulation can be used to explore natural phenomena (e.g., flocking behaviors, queueing, life cycles).	Algorithms and Programming	Recognizing and Defining Computational Problems
9–10	3A-A-6-12	Use a systematic approach and debugging tools to independently debug a program (e.g., setting breakpoints, inspecting variables with a debugger).	Algorithms and Programming	Testing and Refining

INTERIM 2016 CSTA K–12 CS Standards
Level 3A (Grades 9–10)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
9–10	3A-C-7-13	Develop and apply criteria (e.g., power consumption, processing speed, storage space, battery life, cost, operating system) for evaluating a computer system for a given purpose (e.g., system specification needed to run a game, web browsing, graphic design or video editing).	Computing Systems	Communicating About Computing
9–10	3A-C-5-14	Create, extend, or modify existing programs to add new features and behaviors using different forms of inputs and outputs (e.g., inputs such as sensors, mouse clicks, data sets; outputs such as text, graphics, sounds).	Computing Systems	Creating Computational Artifacts
9–10	3A-C-4-15	Demonstrate the role and interaction of a computer embedded within a physical system, such as a consumer electronic, biological system, or vehicle, by creating a diagram, model, simulation, or prototype.	Computing Systems	Developing and Using Abstractions
9–10	3A-C-4-16	Describe the steps necessary for a computer to execute high-level source code (e.g., compilation to machine language, interpretation, fetch-decode-execute cycle). [https://www.cise.ufl.edu/~mssz/CompOrg/CDAintro.html]	Computing Systems	Developing and Using Abstractions
9–10	3A-D-5-17	Create computational models that simulate real-world systems (e.g., ecosystems, epidemics, spread of ideas).	Data and Analysis	Creating Computational Artifacts
9–10	3A-D-4-18	Convert between binary, decimal, and hexadecimal representations of data (e.g., convert hexadecimal color codes to decimal percentages, ASCII/Unicode representation).	Data and Analysis	Developing and Using Abstractions

INTERIM 2016 CSTA K–12 CS Standards
Level 3A (Grades 9–10)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
9–10	3A-D-4-19	Analyze the representation tradeoffs among various forms of digital information (e.g., lossy versus lossless compression, encrypted vs. unencrypted, various image representations).	Data and Analysis	Developing and Using Abstractions
9–10	3A-D-3-20	Discuss techniques used to store, process, and retrieve different amounts of information (e.g., files, databases, data warehouses).	Data and Analysis	Recognizing and Defining Computational Problems
9–10	3A-D-3-21	Apply basic techniques for locating and collecting small- and large-scale data sets (e.g., creating and distributing user surveys, accessing real-world data sets).	Data and Analysis	Recognizing and Defining Computational Problems
9–10	3A-I-2-22	Debate the social and economic implications associated with ethical and unethical computing practices (e.g., intellectual property rights, hacktivism, software piracy, diesel emissions testing scandal, new computers shipped with malware).	Impacts of Computing	Collaborating
9–10	3A-I-7-23	Compare and contrast information access and distribution rights.	Impacts of Computing	Communicating about Computing
9–10	3A-I-7-24	Discuss implications of the collection and large-scale analysis of information about individuals (e.g., how businesses, social media, and government collect and use personal data).	Impacts of Computing	Communicating about Computing

INTERIM 2016 CSTA K–12 CS Standards
Level 3A (Grades 9–10)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
9–10	3A-I-7-25	Describe how computation shares features with art and music by translating human intention into an artifact.	Impacts of Computing	Communicating about Computing
9–10	3A-I-1-26	Compare and debate the positive and negative impacts of computing on behavior and culture (e.g., evolution from hitchhiking to ridesharing apps, online accommodation rental services).	Impacts of Computing	Fostering an Inclusive Computing Culture
9–10	3A-I-1-27	Demonstrate how computing enables new forms of experience, expression, communication, and collaborating.	Impacts of Computing	Fostering an Inclusive Computing Culture
9–10	3A-I-1-28	Explain the impact of the digital divide (i.e., uneven access to computing, computing education, and interfaces) on access to critical information.	Impacts of Computing	Fostering an Inclusive Computing Culture
9–10	3A-I-6-29	Redesign user interfaces (e.g., webpages, mobile applications, animations) to be more inclusive, accessible, and minimizing the impact of the designer's inherent bias.	Impacts of Computing	Testing and Refining
9–10	3A-N-7-30	Describe key protocols and underlying processes of Internet-based services (e.g., http/https and SMTP/IMAP, routing protocols).	Networks and the Internet	Communicating about Computing
9–10	3A-N-4-31	Illustrate the basic components of computer networks (e.g., draw logical and topological diagrams of networks including routers, switches, servers, and end user devices; create model with string and paper).	Networks and the Internet	Developing and Using Abstractions

INTERIM 2016 CSTA K–12 CS Standards
Level 3A (Grades 9–10)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
9–10	3A-N-1-32	Compare and contrast multiple viewpoints on cybersecurity (e.g., from the perspective of security experts, privacy advocates, the government).	Networks and the Internet	Fostering an Inclusive Computing Culture
9–10	3A-N-3-33	Explain the principles of information security (confidentiality, integrity, availability) and authentication techniques.	Networks and the Internet	Recognizing and Defining Computational Problems
9–10	3A-N-3-34	Use simple encryption and decryption algorithms to transmit/receive an encrypted message.	Networks and the Internet	Recognizing and Defining Computational Problems
9–10	3A-N-6-35	Identify digital and physical strategies to secure networks and discuss the tradeoffs between ease of access and need for security.	Networks and the Internet	Testing and Refining

INTERIM 2016 CSTA K–12 CS Standards
Level 3B (Grades 11–12)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
11–12	3B-A-2-1	Use version control systems, integrated development environments (IDEs), and collaborating tools and practices (code documentation) in a group software project.	Algorithms and Programming	Collaborating
11–12	3B-A-2-2	Demonstrate software life cycle processes (e.g., spiral, waterfall) by participating on software project teams (e.g., community service project with real-world clients).	Algorithms and Programming	Collaborating
11–12	3B-A-7-3	Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).	Algorithms and Programming	Communicating about Computing
11–12	3B-A-7-4	Explain security issues that might lead to compromised computer programs (e.g., circular references, ambiguous program calls, lack of error checking and field size checking).	Algorithms and Programming	Communicating about Computing
11–12	3B-A-7-5	Compare a variety of programming languages and identify features that make them useful for solving different types of problems and developing different kinds of systems (e.g., declarative, logic, parallel, functional, compiled, interpreted, real-time).	Algorithms and Programming	Communicating about Computing
11–12	3B-A-7-6	Describe how artificial intelligence drives many software and physical systems (e.g., autonomous robots, computer vision, pattern recognition, text analysis).	Algorithms and Programming	Communicating about Computing
11–12	3B-A-5-7	Decompose a problem by creating new data types, functions, or classes.	Algorithms and Programming	Creating Computational Artifacts
11–12	3B-A-5-8	Demonstrate code reuse by creating programming solutions using libraries and APIs (e.g., graphics libraries, maps API).	Algorithms and Programming	Creating Computational Artifacts

INTERIM 2016 CSTA K–12 CS Standards
Level 3B (Grades 11–12)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
11–12	3B-A-5-9	Implement an AI algorithm to play a game against a human opponent or solve a problem.	Algorithms and Programming	Creating Computational Artifacts
11–12	3B-A-5-10	Develop programs for multiple computing platforms (e.g., computer desktop, web, mobile).	Algorithms and Programming	Creating Computational Artifacts
11–12	3B-A-4-11	Critically analyze classic algorithms (e.g., sorting, searching) and use in different contexts, adapting as appropriate.	Algorithms and Programming	Developing and Using Abstractions
11–12	3B-A-4-12	Evaluate algorithms (e.g., sorting, searching) in terms of their efficiency, correctness, and clarity.	Algorithms and Programming	Developing and Using Abstractions
11–12	3B-A-4-13	Compare and contrast fundamental data structures and their uses (e.g., lists, maps, arrays, stacks, queues, trees, graphs).	Algorithms and Programming	Developing and Using Abstractions
11–12	3B-A-4-14	Discuss issues that arise when breaking large-scale problems down into parts that must be processed simultaneously on separate systems (e.g., cloud computing, parallelization, concurrency).	Algorithms and Programming	Developing and Using Abstractions
11–12	3B-A-3-15	Provide examples of computationally solvable problems and difficult-to-solve problems.	Algorithms and Programming	Recognizing and Defining Computational Problems

INTERIM 2016 CSTA K–12 CS Standards
Level 3B (Grades 11–12)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
11–12	3B-A-3-16	Explain the value of heuristic algorithms (discovery methods) to approximating solutions for difficult-to-solve computational problems.	Algorithms and Programming	Recognizing and Defining Computational Problems
11–12	3B-A-3-17	Decompose a large-scale computational problem by identifying generalizable patterns and applying them in a solution.	Algorithms and Programming	Recognizing and Defining Computational Problems
11–12	3B-A-3-18	Illustrate the flow of execution of a recursive algorithm.	Algorithms and Programming	Recognizing and Defining Computational Problems
11–12	3B-A-3-19	Describe how parallel processing can be used to solve large problems (e.g., SETI at Home, FoldIt).	Algorithms and Programming	Recognizing and Defining Computational Problems
11–12	3B-A-3-20	Develop and use a series of test cases to verify that a program performs according to its design specifications.	Algorithms and Programming	Recognizing and Defining Computational Problems
11–12	3B-A-6-21	Evaluate key qualities of a program (e.g., correctness, usability, readability, efficiency, portability, scalability) through a process such as a code review.	Algorithms and Programming	Testing & Iterative Refinement

INTERIM 2016 CSTA K–12 CS Standards
Level 3B (Grades 11–12)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
11–12	3B-C-7-22	Explain the role of operating systems (e.g., how programs are stored in memory, how data is organized/retrieved, how processes are managed and multi-tasked).	Computing Systems	Communicating about Computing
11–12	3B-C-7-23	Identify the functionality of various categories of hardware components and communication between them (e.g., physical layers, logic gates, chips, input and output devices).	Computing Systems	Communicating about Computing
11–12	3B-D-4-24	Use data analysis to identify significant patterns in complex systems (e.g., take existing data sets and make sense of them).	Data and Analysis	Developing and Using Abstractions
11–12	3B-D-4-25	Discuss how data sequences (e.g., binary, hexadecimal, octal) can be interpreted in a variety of forms (e.g., instructions, numbers, text, sound, image).	Data and Analysis	Developing and Using Abstractions
11–12	3B-D-4-26	Evaluate the ability of models and simulations to formulate, refine, and test hypotheses.	Data and Analysis	Developing and Using Abstractions
11–12	3B-D-4-27	Identify mathematical and computational patterns through modeling and simulation (e.g., regression, Runge-Kutta, queueing theory, discrete event simulation).	Data and Analysis	Developing and Using Abstractions
11–12	3B-D-1-28	Use various data collection techniques for different types of problems (e.g., mobile device GPS, user surveys, embedded system sensors, open data sets, social media data sets).	Data and Analysis	Fostering an Inclusive Computing Culture
11–12	3B-D-3-29	Explore security policies by implementing and comparing encryption and authentication strategies (e.g., secure coding, safeguarding keys).	Data and Analysis	Recognizing and Defining Computational Problems

INTERIM 2016 CSTA K–12 CS Standards
Level 3B (Grades 11–12)

The 2011 CSTA K–12 CS Standards were categorized into five conceptual strands (Computational Thinking, Collaboration, Computing Practice & Programming, Computer & Communication Devices; and Community, Global & Ethical Impacts). The Interim 2016 CSTA K–12 CS Standards are categorized into five concepts of the K–12 CS Framework, which is currently under development (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing). There is some overlap between strands and concepts, but they are not identical.

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
11–12	3B-I-7-30	Develop criteria to evaluate the beneficial and harmful effects of computing innovations on people and society.	Impacts of Computing	Communicating about Computing
11–12	3B-I-5-31	Select, observe, and contribute to global collaboration in the development of a computational artifact (e.g., contribute the resolution of a bug in an open-source project hosted on GitHub).	Impacts of Computing	Creating Computational Artifacts
11–12	3B-I-1-32	Design and implement a study that evaluates or predicts how computation has revolutionized an aspect of our culture and how it might evolve (e.g., education, healthcare, art/entertainment, energy).	Impacts of Computing	Fostering an Inclusive Computing Culture
11–12	3B-I-1-33	Debate laws and regulations that impact the development and use of software.	Impacts of Computing	Fostering an Inclusive Computing Culture
11–12	3B-I-1-34	Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society.	Impacts of Computing	Fostering an Inclusive Computing Culture
11–12	3B-N-4-35	Simulate and discuss the issues (e.g., bandwidth, load, delay, topology) that impact network functionality (e.g., use free network simulators).	Networks and the Internet	Developing and Using Abstractions

APPENDIX A: GLOSSARY OF TERMS

K–12 CS Framework Draft (6/7/2016) Glossary

The following *draft* glossary includes definitions of terms used in the statements in the K–12 CS Framework (https://k12cs.org/). 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. CSTA would like to extend a heartfelt thank you to the K-12 CS Framework for allowing us to include this glossary as part of our supplemental materials.

Term	Definition
abstraction	*(process):* The process of reducing complexity by focusing on the main idea. By hiding details irrelevant to the question at hand and bringing together related and useful details, abstraction reduces complexity and allows one to focus on the problem. In elementary classrooms, abstraction is hiding unnecessary details to make it easier to think about a problem. *(product):* A new representation of a thing, a system, or a problem that helpfully reframes a problem by hiding details irrelevant to the question at hand. [MA-DLCS]
algorithm	A step-by-step process to complete a task.
app	A type of application software designed to run on a mobile device, such as a smartphone or tablet computer (also known as a mobile application). [Techopedia]
application programming interface (API)	A software program that facilitates interaction with other software programs. It allows a programmer to interact with an application using a collection of callable functions, and to write programs that will not cease to function if the underlying system is upgraded. [Techopedia]
artifact	Anything created by a human. *See "computational artifact" for the computer science-specific definition.*
audience	Expected end users of a computing artifact or system.
automate; automation	**automate**: To link disparate systems and software in such a way that they become self-acting or self-regulating. **automation**: The process of automating.
backup	The process of making copies of data or data files to use in the event the original data or data files are lost or destroyed. [Techopedia]
binary	A method of encoding data using two symbols (usually 1 and 0). To illustrate binary encoding, we can use any two symbols. [MA-DLCS]
Bluetooth	Wireless technology that enables communication between Bluetooth-compatible devices. For example, it is used for short-range connections between desktop and laptop computers, digital cameras, scanners, cellular phones, and printers.
bug	An error in a software program. It may cause a program to unexpectedly quit or behave in an unintended manner. [TechTerms] The process of removing errors (bugs) is called debugging.
civic virtues	Principles and traits of character that enable citizens to contribute to the common good by engaging in political and civil society. Reference: C3 Framework for Social Studies.
cloud	Remote servers that store data and are accessed from the Internet. [Techopedia]
code	Any set of instructions expressed in a programming language. [MA-DLCS]

Term	Definition
command	A specific action assigned to a program to perform a specific task. [Techopedia]
compatibility; compatible	The capacity for two systems to work together without having to be altered to do so. Compatibility can refer to interoperability between any two products: hardware and/or software, products of the same or different types, or different versions of the same product. [TechTarget]
complexity	The intrinsic minimum amount of resources, for instance, memory, time, messages, etc., needed to solve a problem or execute an algorithm. [NIST/DADS]
component	An element of a larger group. Usually, a component provides a particular function or group of related functions. [TechTerms, TechTarget]
computational	Relating to computers or computing methods.
computational artifact	An invention, creation, final product, or development by-product, created by the act or process of computing. Often, this term refers to a program. [MA-DLCS, CSP, University of Rhode Island]
computational thinking	The thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent (for example, a computer) [Cuny, Snyder, & Wing, 2010]
computer	A machine or device that performs processes, calculations and operations based on instructions provided by a software or hardware program. [Techopedia]
computer science	The study of computers and algorithmic processes, including their principles, design, implementation, and impact on society. [MA-DLCS, CSTA]
computing	Any goal-oriented activity requiring, benefiting from, or creating algorithmic processes. [MA-DLCS]
computing device	A physical device, although not necessarily in the form of a traditional computer, that performs the functions of a computer. Like a computer, a computing device 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 computing system consists of one or more computers or computing devices, together with their hardware and software. 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; conditional statement	A feature of a programming language that performs different computations or actions depending on whether a programmer-specified Boolean condition evaluates to true or false. [MA-DLCS]
configuration	*(process):* Defining the options that are provided when installing or modifying hardware and software, or the process of creating the configuration (product). [TechTarget] *(product):* The specific hardware and software details that tell exactly what the system is made up of, especially in terms of devices attached, capacity or capability. [TechTarget]
connection	A physical or wireless attachment between multiple computing systems, computers, or computing devices.
connectivity	A program or device's ability to link with other programs and devices. [Webopedia]
control; control structure	**control:** *(in general)* The power to direct the course of actions. *(in programming)* The means of directing which actions take place, and the order in which they take place, implemented through elements of programming code. **control structure:** A programming structure that implements control.

Term	Definition
cultural practices	The manifestations of culture or sub-culture, especially in regard to the traditional and customary practices of a particular ethnic or other cultural group.
data	Information that is 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. [CAS-Prim, TechTerms]
data structure	A particular way to store and organize data within a computer program. [MA-DLCS]
data transmission	The process of sending digital or analog data over a communication medium to one or more computing, network, communication or electronic devices. [Techopedia]
data type	An attribute that tells what kind of data a value or variable can have, as well as what types of operations can be performed on it. [Techopedia, Wikipedia]
database	An organized collection of data, an electronic system that allows data to be easily accessed, manipulated and updated. [Techopedia]
debugging	The process of finding and correcting errors (bugs) in programs. [MA-DLCS]
decomposition; decomposed	**decomposition**: Breaking down a problem or system into its components. [MA-DLCS] **decomposed**: Broken down into components.
device	A unit of physical hardware or equipment that provides one or more computing functions within a computer system. It can provide input to the computer, accept output, or both. [Techopedia]
digital	A characteristic of electronic technology that uses discrete values, generally 0 and 1, to generate, store, and process data. [Techopedia]
digital citizenship	The norms of appropriate, responsible behavior with regard to the use of technology. [MA-DLCS]
digital divide	The gap between those who have access to digital technology and those who do not, which is influenced by social, cultural and economic factors. [MA-DLCS]
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, such as comparisons or data moves, the memory used, the number of messages passed, the number of disk accesses, etc. [NIST/DADS]
encryption	The conversion of electronic data into another form, called cipher text, which cannot be easily understood by anyone except authorized parties. [TechTarget]
end user (or user)	A person for whom a hardware or software product is designed (as distinguished from the developers, installers, and servicers of the product). [TechTarget]
event	Any identifiable occurrence that has significance for system hardware or software. User-generated events include keystrokes and mouse clicks; system-generated events include program loading and errors. [TechTarget]
execute; execution; executable	**execute:** To carry out (or "run") an instruction or instruction set (program, app, etc.) **execution:** The process of executing an instruction or instruction set. **executable:** A binary file containing a program in machine language which is ready to be executed. [FOLDOC]
firewall	A network security system with rules to control incoming and outgoing traffic. [MA-DLCS]
function	A type of procedure or routine. Some programming languages make a distinction between a function, which returns a value, and a procedure, which performs some operation, but does not return a value. [MA-DLCS] *Note: This definition differs from that used in math.*

Term	Definition
functional programming	A programming paradigm—a style of building the structure and elements of computer programs—that treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data. [Wikipedia] Functional programming languages rely heavily on recursion, using it where a procedural language would use looping. [FOLDOC]
GPS	Abbreviation for "Global Positioning System." GPS is a satellite navigation system used to determine the ground position of an object. [TechTerms]
hacking	Appropriately applying ingenuity (from “The Meaning of Hack”), cleverly solving a programming problem (the New Hacker’s Dictionary), and using a computer to gain unauthorized access to data within a system. [MA-DLCS]
hardware	The physical components that make up a computing system, computer, or computing device. [MA-DLCS]
hierarchy	An organizational structure in which items are ranked according to levels of importance. [TechTarget]
human-computer interaction (HCI)	The study of how people interact with computers and to what extent computing systems are or are not developed for successful interaction with human beings. [TechTarget]
identifier	The user-defined, unique name of a program element (such as a variable or procedure) in code. An identifier name should indicate the meaning and usage of the element being referred. [Techopedia]
input	The signals or instructions sent to a computer. [Techopedia]
Internet	The global collection of computer networks and their connections, all using shared protocols to communicate [CAS-Prim]
iterative	Involving the repeating of a process with the aim of approaching a desired goal, target, or result. [MA-DLCS]
logic (Boolean)	Boolean logic deals with the basic operations of truth values: AND, OR, NOT and combinations thereof. [FOLDOC]
loop; looping	**loop**: A programming structure that repeats a sequence of instructions as long as a specific condition is true. [TechTerms] **looping**: Repetition, using a loop.
memory	Temporary storage used by computing devices. [MA-DLCS]
model	A representation of (some part of) a problem or a system. (Modeling (v): the act of creating a model) [MA-DLCS] *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 being recombined in a new application.
network	A group of computing devices (personal computers, phones, servers, switches, routers, and so on) connected by cables or wireless media for the exchange of information and resources.
operating system	Software that communicates with the hardware and allows other programs to run. An operating system (or “OS”) is comprised of system software, or the fundamental files a computer needs to boot up and function. Every desktop computer, tablet, and smartphone includes an operating system that provides basic functionality for the device. [TechTerms]
operation	An action, resulting from a single instruction, that changes the state of data. [Dictionary.com]

Term	Definition
pair programming	A technique in which two developers (or students) team together and work on one computer. [TechTarget] The terms "driver" and "navigator" are often used for the two roles. In a classroom setting, teachers often specify that students switch roles frequently (or within a specific period of time).
paradigm (programming)	A theory or a group of ideas about how something should be done, made, or thought about. A philosophical or theoretical framework of any kind. [Merriam-Webster] Common programming paradigms are object-oriented, functional, imperative, declarative, procedural, logic, and symbolic. [DC, Wikipedia]
parameter	A special kind of variable used in a procedure to refer to one of the pieces of data provided as input to the procedure. These pieces of data are called arguments. An ordered list of parameters is usually included in the definition of a subroutine so each time the subroutine is called, its arguments for that call can be assigned to the corresponding parameters. [MA-DLCS]
piracy	The illegal copying, distribution, or use of software. [TechTarget]
procedure	An independent code module that fulfills some concrete task and is referenced within a larger body of source code. This kind of code item can also be called a function or a subroutine. 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. A procedure may also be referred to as a function, subroutine, routine, method or subprogram. [Techopedia]
processor	The hardware within a computer or device that executes a program. The CPU (central processing unit) is often referred to as the brain of a computer.
program; programming	**program** *(n)*: A set of instructions that the computer executes in order to achieve a particular objective. [MA-DLCS] **program** *(v)*: To produce a program by programming. **programming**: The craft of analyzing problems and designing, writing, testing, and maintaining programs to solve them. [MA-DLCS]
protocol	The special set of rules that end points in a telecommunication connection use when they communicate. Protocols specify interactions between the communicating entities. [TechTarget]
prototype; prototyping	**prototype**: An early approximation of a final product or information system, often built for demonstration purposes. [TechTarget, Techopedia] **prototyping**: The process of creating a prototype.
pseudocode	A detailed yet readable description of what a computer program or algorithm must do, expressed in a formally-styled natural language rather than in a programming language. [TechTarget]
recursion; recursive function	**recursion**: An algorithmic technique in which a function, in order to accomplish a task, calls itself with some part of the task. [NIST/DADS] **recursive function**: A function, implemented in a programming language, that calls (invokes) itself. [MA-DLCS, Techopedia]
redundancy	A system design in which a component is duplicated so if it fails there will be a backup. [TechTarget]
reliability	An attribute of any system that consistently produces the same results, preferably meeting or exceeding its specifications. [FOLDOC]
routing; router	**routing:** Establishing the path that data packets traverse from source to destination. **router:** A device or software that determines the routing for a data packet. [TechTarget]

Term	Definition
security	The protection against access to, or alteration of, computing resources, through the use of technology, processes, and training. [TechTarget]
simulate; simulation	**simulate**: to imitate the operation of a real world process or system over time. **simulation**: Imitation of the operation of a real world process or system over time. [MA-DLCS]
software	Programs that run on a computer system, computer, or other computing device.
storage	(1) A place (usually a device) into which data can be entered, in which it can be held, and from which it can be retrieved at a later time. [FOLDOC] (2) A process through which digital data is saved within a data storage device by means of computing technology. Storage is a mechanism that enables a computer to retain data, either temporarily or permanently. [Techopedia]
string	A sequence of letters, numbers, and/or other symbols. A string might represent a name, address, or song title. Some functions commonly associated with strings are length, concatenation, and substring. [TechTarget]
structure	A general term used in the framework to discuss the concept of encapsulation without specifying a particular paradigm.
subroutine	A callable unit of code, a type of procedure.
switching; switch	**switching:** The practice of directing a signal or data element toward a particular hardware destination. [Techopedia] **switch:** A high-speed device that receives incoming data packets and redirects them to their destination on a local area network (LAN). [Techopedia]
syncing	Merging data from multiple computing systems, computers, or computing devices.
syntax	The grammar, structure, or order of the elements in a programming language statement. [TechTarget]
system; systems thinking	**system**: A collection of elements or components that work together for a common purpose. [TechTarget] **(computing) system**: A collection of computing hardware and software integrated for the purpose of accomplishing shared tasks. **systems thinking:** A holistic approach to analysis that focuses on the way that a system's constituent parts interrelate and how systems work over time and within the context of larger systems. [TechTarget]
	The physical and logical configuration of a network; the arrangement of a network, including its nodes and connecting links. A logical topology is how devices appear connected to the user. A physical topology is how they are actually interconnected with wires and cables. [PC Magazine]
topology	The physical and logical configuration of a network; the arrangement of a network, including its nodes and connecting links. A logical topology is how devices appear connected to the user. A physical topology is how they are actually interconnected with wires and cables. [PC Magazine]
troubleshooting	A systematic approach to problem solving that is often used to find and resolve a problem, error, or fault within software or a computer system. [Techopedia, TechTarget]
USB	Abbreviation for "Universal Serial Bus." USB is the most common type of computer port used in today's computers. It can be used to connect keyboards, mice, game controllers, printers, scanners, digital cameras, and removable media drives, just to name a few. [TechTalk]
user	*See the definition for "end user."*
variable	A symbolic name that is used to keep track of a value that can change while a program is running. Variables are not just used for numbers. They can also hold text, including whole

	sentences (“strings”), or the logical values “true” or “false.” A variable has a data type and is associated with a data storage location; its value is normally changed during the course of program execution. [CAS-Prim, Techopedia] *Note: This definition differs from that used in math.*

Draft Version 06/07/2016

Key to sources of multiple definitions in this glossary:

CAS-Prim	Computing At School. Computing in the national curriculum: A guide for primary teachers (http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf)
CSTA	CSTA K-12 Computer Science Standards (2011) https://csta.acm.org/Curriculum/sub/K12Standards.html
FOLDOC	Free On-Line Dictionary of Computing. (http://foldoc.org/)
MA-DLCS	Massachusetts Digital Literacy and Computer Science (DL&CS) Standards, Glossary (Draft, December 2015)
NIST/DADS	National Institute of Science and Technology Dictionary of Algorithms and Data Structures. (https://xlinux.nist.gov/dads//)
Techopedia	Techopedia. (https://www.techopedia.com/dictionary)
TechTarget	TechTarget Network. (http://www.techtarget.com/network)
TechTerms	Tech Terms Computer Dictionary. (http://www.techterms.com)

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

A few notes regarding this glossary:

- The first source was the draft Massachusetts glossary for their digital literacy and computer science standards.
- We did not define any words in which the definition is the same in common English.
- We only define terms that are used in the framework (the statements themselves, the subconcept headings, the core concept headings, core practice headings, the descriptive material/elaboration). After each draft of the framework, we check that there are no terms in the glossary that are no longer used in the framework.
- You'll notice many of the definitions include a source. The sources are all described at the bottom of the doc. Most of the definitions are rephrased from the source, but some are taken almost word-for-word. Others are combinations of pieces of definitions from multiple sources. And some of the definitions are simply from the heads of the writers.

APPENDIX B: LEGEND FOR IDENTIFIERS

Unique Numbering System for the 2016 CSTA K–12 CS Standards

To help organize and track each individual standard, we have developed a unique identifier for each standard. An example appears below:

Grades	Identifier	Interim CSTA K–12 CS Standard	Framework Concept	Framework Practice
9–10	3A-A-2-1	Design and develop a software artifact working in a team.	Algorithms and Programming	Collaborating

Use the following legend to interpret the unique identifier for each [Interim] K-12 CS Standard:

The identifier code corresponds to: **Level – Concept – Practice – Identifier**		
Identifier Code		**Key**
Levels	1A	Grades K–2
	1B	Grades 3–5
	2	Grades 6–8
	3A	Grades 9–12
	3B	Grades 11–12
Concepts	I	Impacts of Computing
	A	Algorithms and Programming
	D	Data and Analysis
	N	Networks and the Internet
	C	Computing Systems
Practices	1	Fostering an Inclusive Computing Culture
	2	Collaborating
	3	Recognizing and Defining Computational Problems
	4	Developing and Using Abstractions
	5	Creating Computational Artifacts
	6	Testing and Refining
	7	Communicating about Computing

CSTEACHERS.ORG
COMPUTER SCIENCE TEACHERS ASSOCIATION

Association for Computing Machinery