K-12 Louisiana Student Standards for Computer Science

Table of Contents

Updated May 20, 2025

Introduction

Development of K-12 Louisiana Student Standards for Computer Science

Louisiana Act 541 (2022), the Computer Science Education Act, aimed to establish a comprehensive and seamless statewide computer science education program across all educational levels to meet the demands of a dynamic economy. To achieve this goal, Act 541 created the Computer Science Education Advisory Commission (CSEAC). CSEAC was responsible for advising the State Board of Elementary and Secondary Education (BESE) through the state Department of Education (LDOE) to develop and implement a state action plan for delivering education in computer science in all public schools.

On October 11, 2023, the State Board of Elementary and Secondary Education (BESE) received the formal report from the CSEAC and subsequently directed the Louisiana Department of Education (LDOE) to develop a comprehensive set of K-12 computer science content standards. Upon BESE approval of the timeline, process, and overall structure of the K-12 Computer Standards Writing Committee, members were selected through a competitive application process. The committee, comprised of diverse stakeholders, such as classroom educators, school system leaders, business and industry representatives, higher education faculty, and parents, ensured a collaborative and comprehensive approach to the standards development process.

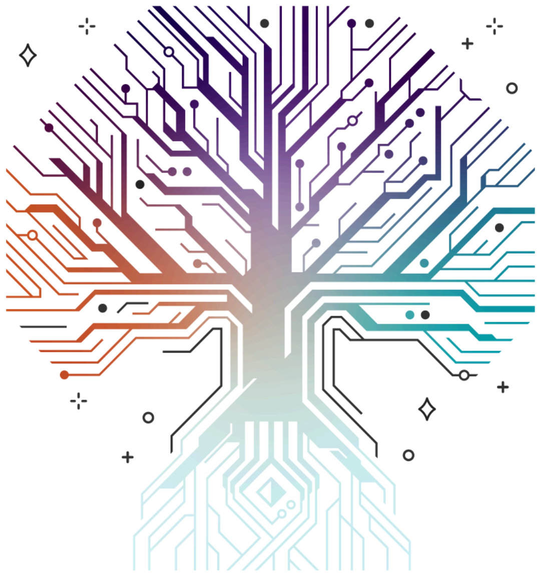
From May to August 2024, the K-12 Computer Standards Writing Committee drafted student-centered and developmentally appropriate computer science standards within defined grade bands: elementary (K–5), middle school (6–8), and high school (9–12). These standards were carefully designed to reflect the increasing complexity and depth of understanding expected as students progress through each educational stage, with a focus on analytical and critical thinking skills, digital literacy, digital citizenship, and technology skills.

The proposed standards underwent public review and comment, allowing stakeholders the opportunity to provide feedback. In October 2024, BESE received the K-12 Louisiana Student Standards for Computer Science, which now serve as the guiding framework for high-quality computer science instruction across the state, ensuring Louisiana students are prepared to excel in the digital age.

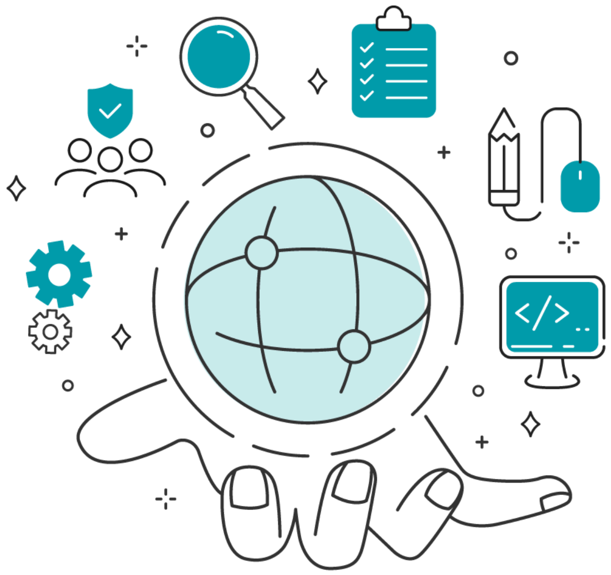
Louisiana Vision for Computer Science Education

Louisiana's vision for K-12 Computer Science Education is to increase **digital literacy** skills through engagement with a **progression of computer science concepts and experiences** which prepare all students for success in society and future **career opportunities.**

Digital Literacy



Progression of Computer Science Concepts and Experiences



Career Opportunities

Louisiana Computer Science Framework

The Louisiana Computer Science Framework (LCSF) provides a structured learning progression within the standards, ensuring students build foundational knowledge and essential skills over time. It outlines core concepts and practices that help students understand key ideas in computer science, develop problem-solving abilities, and apply computational thinking.

Core Concepts

Core concepts represent fundamental ideas in computer science that students explore throughout their education and enable students to develop a comprehensive knowledge of computer science. The standards are organized around these five core concepts.

Core Practices

Core practices encompass the skills and habits of mind that students develop while learning computer science. These seven practices are interconnected and reinforce one another, guiding students in analyzing problems, designing solutions, and communicating their understanding.

Core Concepts	Core Practices
1. Computing Systems	1. Fostering responsible cyber citizenship
2. Networks and the Internet	2. Collaborating around computing
3. Data and Analysis	3. Recognizing and defining computational problems
4. Algorithms and Programming	4. Developing and using abstractions
5. Impacts of Computing	5. Creating computational artifacts
	6. Testing and refining computational artifacts
	7. Communicating about computing

Structure and Components of the Standards

Organization by Grade Bands

The computer science standards are structured into three distinct grade bands: elementary (kindergarten–grade 5), middle (grade 6–grade 8), and high (grade 9–grade 12). This organization ensures that students are introduced to foundational concepts early and build upon them with increasing complexity and depth as they advance through each grade level.

- In elementary grades, students are introduced to basic computer science concepts through age-appropriate activities. Students begin developing a foundational understanding of computing systems, algorithms, data, and digital citizenship. The focus is on exploration, vocabulary development, and simple problem-solving using computational thinking.

- In middle school grades, students expand on these foundations with more complex applications. Students engage in more structured problem-solving, use programming to create functional solutions, and explore concepts such as networks, cybersecurity, and data analysis more deeply. The emphasis shifts toward applying foundational skills to real-world scenarios and collaborative projects.

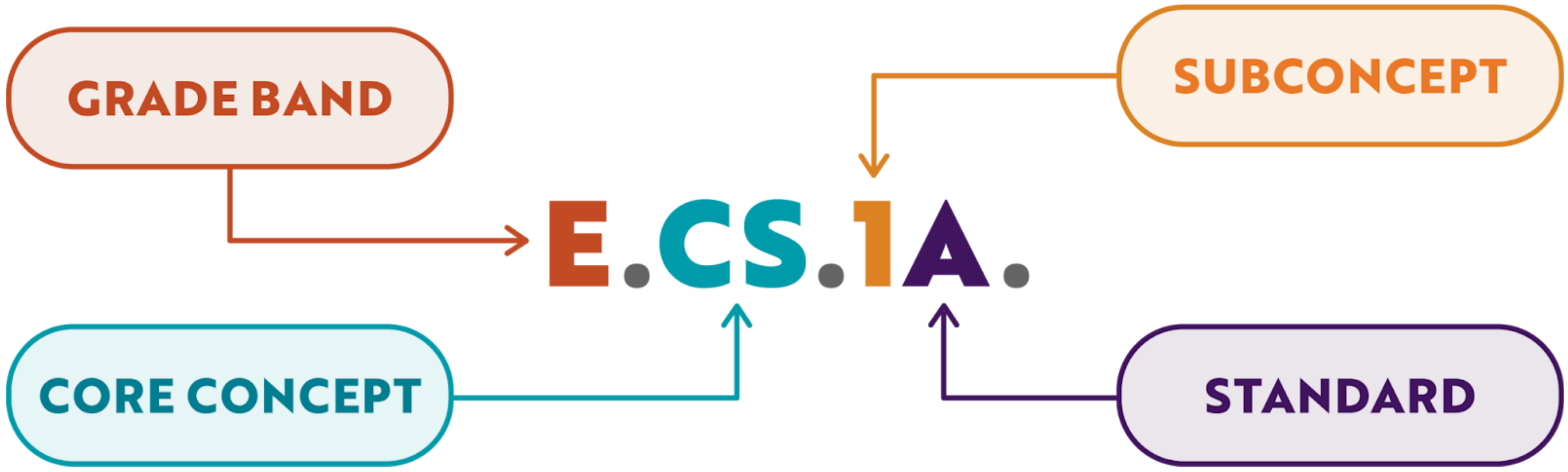
- In high school grades, students learn the most advanced topics, exploring theoretical and practical aspects of computer science. Students design and implement complex programs, analyze algorithms, study data structures, and investigate the societal impacts of technology. Students are encouraged to think critically and creatively as they solve intricate problems and prepare for potential further education or careers in the computer science field.

This progression across grade bands ensures a developmentally appropriate learning experience in which each core concept is revisited with increasing sophistication and depth, allowing students to develop a strong and comprehensive understanding of computer science over time.

Standards Coding

The coding system for the computer science standards is designed to be clear and systematic, allowing easy identification of the grade band, core concept, subconcept, and specific standard. The first letter in the code represents the grade band: "E" for elementary, "M" for middle school, and "H" for high school. Following the grade band, the core concept is indicated by a two-letter abbreviation: "CS" stands for Computing Systems, "NI" for Networks and the Internet, "DA" for Data and Analysis, "AP" for Algorithms and Programming, and "IC" for Impacts of Computing. The final part of the code includes a sequential number and a letter that denote specific subconcepts within the core concept.

For instance, in the example code "E.CS.1A," "E" signifies that the standard is from the elementary grade band, and "CS" indicates it pertains to the core concept Computing Systems."1A" specifies with the number that this standard is for the first subconcept under Computing Systems, which is Hardware and Software. The letter “A” indicates that this is the first standard listed under this subconcept. This structured approach ensures that educators can quickly locate and reference specific standards relevant to their teaching needs, providing a consistent and efficient way to navigate the standards.



Louisiana Student Standards for Computer Science by Grade Band

Kindergarten through Grade 5

Core Concept 1: Computing Systems

Hardware and Software

E.CS.1A. Identify and select the appropriate hardware to complete computing tasks.

E.CS.1B. Identify and select the appropriate software to complete computing tasks.

E.CS.1C. Evaluate hardware and software types to meet users' needs in completing various computing tasks.

Troubleshooting

E.CS.2A. Propose potential ways to address computing problems using appropriate hardware or software.

Core Concept 2: Networks and the Internet

Hardware and Network Communication

E.NI.1A. Explain how networks connect computers to other computing systems and the internet.

Cybersecurity

E.NI.2A. Describe personally identifiable information (PII) and identify practices for when and where sharing PII is appropriate.

E.NI.2B. Identify ways to maintain data security when using networks.

Core Concept 3: Data and Analysis

Data Representation

E.DA.1A. Organize and present data visually to highlight relationships and support claims.

E.DA.1B. Classify types of data and describe the attributes used to sort data.

Data Collection

E.DA.2A. Select the appropriate data collection tool and technique to gather data to support a claim or communicate information.

E.DA.2B. Describe and collect data utilizing the appropriate units of measure and discuss how data format impacts a computing system.

Data Storage

E.DA.3A. Compare and contrast ways to store data using technology.

E.DA.3B. Explain how to save and name data, search for data, retrieve data, modify data, and delete data using a computing device.

Visualizations and Transformations

E.DA.4A. Organize and present data visually in at least three ways to highlight relationships and evaluate a claim.

E.DA.4B. Evaluate data quality and clean data when indicated using the criteria of validity, accuracy, completeness, consistency, and uniformity.

Inference and Models

E.DA.5A. Utilize data to create models, answer investigative questions, and make predictions.

E.DA.5B. Analyze data for patterns and relationships.

Core Concept 4: Algorithms and Programming

Variables and Algorithms

E.AP.1A. Create clearly named variables representing different data types and perform operations on the variables’ values.

E.AP.1B. Create, use, and apply an algorithm to complete a task. Compare the results of algorithm usage trials and refine the algorithm.

Control Structures

E.AP.2A. Define what a control structure is and create programs that include sequences, conditionals, events, and loops.

Modularity

E.AP.3A. Define and apply decomposition to a complex problem in order to create smaller subproblems that can be solved through step-by-step instructions.

E.AP.3B. Modify, remix, or incorporate parts of an existing problem's solution to develop something new or add more advanced features to a program.

Program Development

E.AP.4A. Create a simple program to achieve a goal with expected outcomes.

E.AP.4B. Test and debug a program or algorithm to ensure the program produces the intended outcome.

E.AP.4C. Collaborate with a team of peers to design, implement, test, and review the stages of program development.

E.AP.4D. Identify intellectual property rights and apply the appropriate attribution when creating or remixing programs.

E.AP.4E. Describe and justify the steps taken and choices made during a program's development.

E.AP.4F. Using an iterative process, test a program step-by-step and document areas of refinement.

Core Concept 5: Impacts of Computing

Intellectual Achievements

E.IC.1A. Describe how computing has changed the ways people live and work.

Social Interaction

E.IC.2A. Identify and describe examples of appropriate versus inappropriate computer communications.

E.IC.2B. Identify examples of cyberbullying with age-appropriate responses.

Laws, Safety, and Industry Practices

E.IC.3A. Explain how online actions have real-world consequences and that laws and rules may also apply online.

E.IC.3B. Describe the safe versus unsafe uses of computing systems at age-appropriate levels.

E.IC.3C. Explain how the school and school system's computing rules and policies keep students safe.

Grade 6 through Grade 8

Core Concept 1: Computing Systems

Hardware and Software

M.CS.1A. Analyze the functions and interactions of core components within a computer system.

M.CS.1B. Explain how hardware and software components work together to perform specific tasks.

Troubleshooting

M.CS.2A Evaluate possible solutions to a hardware or software problem.

Core Concept 2: Networks and the Internet

Hardware and Network Communication

M.NI.1A. Analyze the various structures and functions of a network.

M.NI.1B. Identify and differentiate the protocols utilized in data sharing across networks.

Cybersecurity

M.NI.2A. Analyze threats and vulnerabilities to information security for individuals and organizations.

M.NI.2B. Explain how physical and digital security practices and measures proactively address threats to users, data, and devices within and across networks.

Core Concept 3: Data and Analysis

Data Representation

M.DA.1A. Analyze and explain the connection between data sets and graphical representations.

M.DA.1B. Evaluate the most efficient and effective ways to arrange, collect, and visually represent data to inform others.

Data Collection

M.DA.2A. Compare and contrast how data is collected using computational and non-computational tools and processes.

M.DA.2B. Analyze scenarios and computing systems to determine the appropriate data entry format for specific tasks.

Data Storage

M.DA.3A. Propose methods to back up data safely and the appropriate practices for data risk management.

M.DA.3B. Describe how different representations of real-world phenomena, such as letters, numbers, and images are encoded as data.

Visualizations and Transformations

M.DA.4A. Utilize tools and techniques to locate, collect, and create visualizations of large-scale data sets.

M.DA.4B. Collect and transform data using computational tools to make functional and reliable data for use in hypothesis testing.

Inference and Models

M.DA.5A. Refine computational models based on data generated by the models.

M.DA.5B. Describe and evaluate the accuracy of a modeled system by comparing the generated results with observed data from the system the data represents.

Core Concept 4: Algorithms and Programming

Variables and Algorithms

M.AP.1A. Evaluate and use naming conventions for variables to accurately communicate the variables' meaning to other users and programmers.

M.AP.1B. Evaluate algorithms in terms of efficiency, correctness, and clarity.

M.AP.1C. Compare and contrast data constants and variables.

Control Structures

M.AP.2A. Explain the functions of various control structures. Compare and contrast examples of control structure types.

M.AP.2B. Design and iteratively develop programs that combine control structures into advanced control structures.

Modularity

M.AP.3A. Decompose problems to facilitate program design, implementation, and review.

M.AP.3B. Create procedures with parameters to organize code and promote reusability.

Program Development

M.AP.4A. Seek and incorporate feedback from peers to employ user-centered design solutions.

M.AP.4B. Use applicable industry practices to test, debug, document, and peer review code.

M.AP.4C. Develop computational artifacts by working as a team, distributing tasks, and maintaining an iterative project timeline.

M.AP.4D. Incorporate existing resources into original programs and give the proper attributions.

M.AP.4E. Systematically test, document outcomes, and refine programs using a range of test cases.

Core Concept 5: Impacts of Computing

Intellectual Achievements

M.IC.1A. Identify foundational computational advancements through the use of the technology innovation cycle.

M.IC.1B. Plan and devise new ideas and solutions for problems with inspiration from previous discoveries in computational knowledge.

Social Interaction

M.IC.2A. Develop and propose norms for informal versus formal online communications.

M.IC.2B. Analyze communication technologies and then describe how the technology's use influences individuals and society.

M.IC.2C. Generate designs that increase the accessibility and usability of technology for various groups of users.

Laws, Safety, and Industry Practices

M.IC.3A. Identify applicable laws that impact personal, industry, or business computing practices.

M.IC.3B. Recommend and propose computing-use guidelines to maintain a user's personal safety, privacy, and well-being.

M.IC.3C. Describe and categorize factors that affect user's access to computing resources locally, nationally, and globally.

Grade 9 through Grade 12

Core Concept 1: Computing Systems

Hardware and Software

H.CS.1A. Analyze the levels of interactions between application software and system software as well as the hardware layers.

H.CS.1B. Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday things.

Troubleshooting

H.CS.2A. Generate guidelines that convey systematic troubleshooting strategies that other users can utilize to identify and fix errors.

Core Concept 2: Networks and the Internet

Hardware and Network Communication

H.NI.1A. Evaluate a network's scalability, reliability, and appropriateness by describing the relationship between routers, switches, devices, topology, and addressing (MAC, IP, Subnet, Gateway).

H.NI.1B. Illustrate how to trace data through a network model, explaining the interactions that occur throughout the process.

H.NI.1C. Describe and evaluate the internet as a digital public infrastructure (DPI) from the highest level to the private service provider level.

Cybersecurity

H.NI.2A. Recommend security measures to address factors that create trade-offs between the usability and security of a computing system.

H.NI.2B. Interpret and analyze mechanisms through which malware and other types of cyber attacks can impact hardware, software, and sensitive data.

H.NI.2C. Compare and contrast how software developers protect computing systems and information from unauthorized user access.

Core Concept 3: Data and Analysis

Data Representation

H.DA.1A. Evaluate data representations, propose strategies to reconstruct the data, and visualize data in a variety of ways.

H.DA.1B. Define and describe database structures to optimize the search and retrieval of data.

Data Collection

H.DA.2A. Explain and describe the impacts of uncertainty and the limitations of data collection technology and tools.

H.DA.2B. Describe the personal and legal impacts of accumulated data, both collected and derived, for given scenarios. Propose tools and techniques to manage the accumulated data appropriately.

Data Storage

H.DA.3A. Justify choices on how data elements are organized and where data is stored considering cost, speed, reliability, accessibility, privacy, and integrity.

H.DA.3B. Explain and utilize the appropriate data structural organization system to collaborate and communicate data within a team or user group in given scenarios.

Visualizations and Transformations

H.DA.4A. Create interactive data visualizations using software tools that explain complex data to others.

H.DA.4B. Utilize data analysis tools to ingest (extract, transform, and load) and process data into relevant information.

Inference and Models

H.DA.5A. Create a model utilizing data with the appropriate simulated variables to develop predictions for real-world phenomena.

H.DA.5B. Apply and evaluate data analysis techniques to identify patterns represented in complex systems.

H.DA.5C. Analyze patterns in data visualizations, then select a collection tool to test a hypothesis and communicate the relevant information to others.

H.DA.5D. Evaluate the impacts of the variables and the model on the performance of a simulation to refine a hypothesis.

Core Concept 4: Algorithms and Programming

Variables and Algorithms

H.AP.1A. Assess variables, then classify the scope and type of variable.

H.AP.1B. Design algorithms that can be adapted to express an idea or solve a problem.

H.AP.1C. Use and adapt classical algorithms to solve computational problems.

H.AP.1D. Explain what computer memory is and how variables are stored and retrieved.

H.AP.1E. Identify and explain how a derived data type can be utilized in a real-world scenario.

Control Structures

H.AP.2A. Justify the selection of control structures to balance implementation complexity, maintainability, and program performance.

H.AP.2B. Design and iteratively develop computational artifacts using events to initiate instructions.

Modularity

H.AP.3A. Decompose problems into smaller components using constructs such as procedures, modules, and/or objects.

H.AP.3B. Create computational artifacts using procedures within a program, combinations of data and procedures, or independent but interrelated programs.

Program Development

H.AP.4A. Utilize the Software Development Life Cycle (SDLC) to create software that is a minimum viable product.

H.AP.4B. Develop and utilize test cases to verify that a program performs according to the program's design specifications.

H.AP.4C. Design and develop programs by working in team roles using version control systems, integrated development environments (IDEs), and collaborative tools and practices.

H.AP.4D. Evaluate licenses that limit or restrict the use of computational artifacts when utilizing resources such as libraries.

H.AP.4E. Apply the appropriate documentation techniques to make programs more accessible to debug and to be maintained by others.

H.AP.4F. Iteratively evaluate and modify an existing program to add functionality and discuss intended and unintended implications.

H.AP.4G. Use a standard library and/or application programming interface (API) to create reusable code components to design simple programs and enhance existing programs.

Core Concept 5: Impacts of Computing

Intellectual Achievements

H.IC.1A. Analyze the key milestones of computer science, historical events influenced by computer science, and the people connected to these achievements.

H.IC.1B. Explain how innovations in computer science and technology enable advancements in other fields of study.

Social Interaction

H.IC.2A. Evaluate the adoption and adaptation of social norms from the physical world to the cyber world.

H.IC.2B. Describe how cyberspace is becoming a universal medium for connecting humans, the economy, business, and computing.

H.IC.2C. Describe and critique how algorithmic feedback loops can shape perceptions, reinforce a limited data set, and limit the sources of information that may inform the individual user.

Laws, Safety, and Industry Practices

H.IC.3A. Debate laws and industry regulations that impact the development and use of computational artifacts.

H.IC.3B. Describe and analyze the motives of online threat actors to a user's personal safety, privacy, and well-being.

H.IC.3C. Compare and contrast the varied approaches to govern data, intellectual property, control information access, and various ways for users to be aware of guidance.

H.IC.3D. Explain how the interconnectedness of cyberspace can lead to physical and digital vulnerabilities.

H.IC.3E. Debate the ethical considerations of creating and publishing computational artifacts.

H.IC.3F. Analyze the data provenance of computational artifacts.

H.IC.3G. Explain how individuals and organizations can exert influence on personal and societal perceptions and practices through computing technologies.

Appendix

Understanding Core Concepts and Subconcepts

Core Concept 1: Computing Systems

Students interact with a wide variety of computers each day. Computers are devices that can collect, store, analyze, and act upon information in ways that can positively and negatively affect human capabilities. Humans (users) operate, program, and maintain the physical components (or hardware) and instructions (or software) that make up a computer. The hardware, software, and users are collectively called computing systems. Users leverage their understanding of hardware and software to resolve problems within computing systems in a process called troubleshooting.

Hardware and Software	Hardware includes the physical components of a computer system (both internal and external) that support key functions such as input, output, processing, storage, and communication. The most common types of software are system software and applications. Early learning focuses on how computing systems use hardware and software to represent and process digital information. Later, students distinguish between types of software, such as applications and system software, and understand how system software manages the flow of information between hardware components through input, output, storage, and processing. Over time, students develop a deeper understanding of how these layers work together to support the functionality of modern computing systems.
Troubleshooting	Troubleshooting strategies assist in identifying and resolving issues when computing systems malfunction or do not perform as expected. Students begin by identifying a problem, which is the essential first step in finding a solution. In the early grades, students are introduced to the basic troubleshooting of common, simple issues. As they progress, students develop more systematic approaches to problem-solving and create their own troubleshooting strategies, informed by a growing understanding of how computing systems function and where issues may arise.

Core Concept 2: Networks and the Internet

Computing systems typically do not operate in isolation. Networks connect computers to share information and resources and are integral to computer and data science. Networks enable critical communication for the computing systems that drive our economy and career sectors. The increased level of connectivity brought about by the internet provides fast and secure communication that facilitates innovation.

Hardware and Network Communication	Computing devices communicate over networks to exchange information. In the early grades, instruction focuses on helping students recognize that computers enable connections with people, places, and digital content. In later grades, students build a more sophisticated understanding of how data is transmitted and received across different networks through specific hardware such as routers, laying the foundation for concepts such as the internet, protocols, and digital communication systems.
Cybersecurity	Sending information securely over networks requires proper safeguards. In the early grades, instruction emphasizes the importance of keeping personal information safe. In later grades, students are introduced to more complex methods for securing data transmission, building foundational knowledge of digital privacy, cybersecurity, and safe online practices.

Core Concept 3: Data and Analysis

Computing systems function by processing and storing data. Data is abundant due to the growing number of connected devices worldwide. As the volume of data expands, so does the demand for accurate and efficient data analysis methods. Data science is the cross-disciplinary use of data to inform decision-making, test hypotheses, predict trends, and develop precise models that drive innovation across industries.

Data Representation	Data representation involves organizing, presenting, classifying, collecting, storing, and visualizing data to support claims, solve problems, and make informed decisions. Students progress from basic data classification and visualization in elementary grades to more complex data management, analysis, and modeling in high school grades, preparing them for real-world applications and future careers.
Data Collection	Data is gathered using computational and non-computational tools and methods. In the early grades, students explore how data about themselves and the world around them is collected and used. This foundational understanding helps students recognize the role of data in everyday digital experiences, forming a foundation for digital awareness. As they advance, students examine the broader impacts of data collection through computational and automated tools, developing an understanding of how data is gathered, analyzed, and used to influence decisions, behaviors, and systems.
Data Storage	Storing, retrieving, and managing data are key functions of computers. Initially, students learn how data is stored on digital devices, building a foundational understanding of storage systems. As they progress, students explore different methods of storing data and compare the features, purposes, and limitations of the methods. Over time, students learn to evaluate storage solutions by considering tradeoffs such as speed, capacity, scalability, accessibility, and security.
Visualizations and Transformations	Data is often transformed to enhance clarity and insight when moving through stages of collection, digital representation, and analysis. In the early grades, students are introduced to simple transformations that make data more understandable. As they advance, students apply more complex operations to organize, analyze, and interpret data, uncovering patterns and trends that help make sense of information and effectively communicate findings to others.

Inferences and Models	Data science is an example of how computer science supports various fields. For instance, computer science and science rely on data to draw inferences, develop theories, or make predictions. In the early grades, students explore how data can be used to make simple predictions. As students progress, they learn how models and simulations help test theories and understand complex systems. Students examine how predictions and inferences evolve when working with larger and more complex data sets.

Core Concept 4: Algorithms and Programming

An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing systems. Algorithms and programs control all computing systems, empowering people to communicate with the world in novel ways and solve compelling problems. The development process to create meaningful and efficient programs involves choosing which information to use, how to process the data, and how to store the information. The decomposition of more significant problems into simpler ones, combined with recombining existing solutions and analyzing various solutions, helps determine the most appropriate solution to a problem.

Variables and Algorithms	Algorithms are structured sets of instructions or procedures designed to be followed systematically. These instructions can be executed by humans, who may manually follow the steps, and computers, which process them automatically. In the early grades, students are introduced to simple, real-world algorithms relevant to everyday experiences to build a foundational understanding of structured thinking. As students advance, they learn how algorithms are developed, combined, and broken down. Students learn how to design, refine, and compare algorithms, evaluating their efficiency and effectiveness in solving different problems. Alongside algorithms, students are introduced to variables (tools that enable computer programs to store, reference, and manipulate data). With experience, students use variables to organize and transform data to solve problems and represent ideas more effectively.
Control Structures	Control structures determine the flow and decision-making within a program or algorithm by defining the order in which instructions are executed. In the early grades, students are introduced to sequential execution to gain an understanding of how computers follow instructions step by step. As students advance, they explore conditional statements, loops, and other control mechanisms that introduce flexibility and complexity into programs, including how combinations of control structures can enable more complex and dynamic execution in programming.

Modularity	Modularity is a software characteristic that emphasizes decomposing a program's functionality so that each module contains everything needed to execute only one aspect of the desired functionality. The modules can be reused in new combinations to make new applications. In the early grades, students learn to decompose problems into smaller steps and reuse known solutions. Similarly, algorithms and programs can be constructed by breaking tasks into smaller steps and reusing existing solutions. As their skills grow, students recognize patterns and apply generalizable strategies to create modular code. Students also develop the ability to communicate their logic clearly, enabling collaboration and knowledge sharing in team-based programming environments.
Program Development	Program development is an iterative process of designing, implementing, debugging, and reviewing code to create satisfactory, functional software. In the early grades, students explore how and why people create programs to solve problems or meet needs. As students progress, they examine the tradeoffs involved in program design, considering factors like user constraints, efficiency, and ethics. Students also learn to test and debug code systematically, ensuring the programs are reliable, efficient, and aligned with user expectations.

Core Concept 5: Impacts of Computing

The impacts of computing can be positive and negative, enabling innovation, communication, and access to information, while also raising concerns around ethics, privacy, and fairness. Individuals and communities not only shape computing systems through interactions, behaviors, industry practices, and laws, but are also shaped by interactions with computing systems. Computer and data science create new means of communication by accelerating information exchange and establishing cyberspace as a dynamic workspace for individuals.

Intellectual Achievements	Computing supports intellectual growth by enabling discoveries, enhancing creativity, and fostering innovation across multiple disciplines. In early grades, students explore how technology transforms daily activities, such as communication and access to information, to recognize computing's broad impact on society. As they advance, students explore the history of technological progress, examining foundational breakthroughs in computing and how each new development built upon previous knowledge. Students also investigate how advances in computing continue to drive progress in other fields, demonstrating the interconnectedness of computer science with disciplines such as medicine, engineering, and the arts.
Social Interaction	Computing provides innovative ways to connect people, communicate information, and express ideas. In the early grades, students learn how digital technologies help connect people and share information. As students progress, they explore how the social aspects of computing influence various communities, institutions, and industries, shaping how people collaborate and interact in personal and professional environments.
Laws, Safety, and Industry Practices	The ethical and legal dimensions of computing are essential to creating a safe and trustworthy digital environment. In the early grades, students learn the basics of digital citizenship, focusing on the responsible and appropriate use of digital media. As they advance, students explore broader topics such as cybersecurity, intellectual property, privacy rights, and regulatory frameworks.

Understanding Core Practices

Core Practice 1: Fostering responsible cyber citizenship

As students engage with digital tools and online spaces, they develop an understanding of what it means to be responsible digital citizens. Responsible digital citizenship involves understanding how to navigate digital spaces thoughtfully, fairly, and safely. Other important aspects of responsible cyber citizenship include safe behavior, respect for others, and protecting personal information. This practice also involves navigating complex issues, such as digital footprints, ethical use of technology, cyberbullying, and the impact of online actions.

Core Practice 2: Collaborating around computing

Students develop collaborative computing skills through group work to solve problems and create digital products. Through work with others, shared responsibilities, peer feedback, and communication, students develop stronger outcomes than those achieved working alone. Collaboration involves navigating diverse viewpoints, resolving disagreements, and contributing unique strengths. Collaborative computing also relies on using digital tools that support coordination, co-creation, and project management, enabling teams of students to build more effective and innovative solutions together.

Core Practice 3: Recognizing and defining computational problems

Computational thinking requires students to clearly define a problem, break it into manageable parts, and evaluate whether computing can be used to solve the problem effectively. This practice includes pattern recognition, designing logical steps, and identifying opportunities where technology can offer meaningful solutions. Recognizing and defining computational problems requires analytical thinking and creative problem-solving to determine when and how computing methods can be used in real-world contexts.

Core Practice 4: Developing and using abstraction

Abstraction is the process of managing complexity by identifying patterns and creating generalizations. Users recognize similarities across different examples and use these insights to develop reusable solutions. By simplifying problems and focusing on essential details, abstraction helps streamline the process and allows for more efficient problem-solving.

Core Practice 5: Creating computational artifacts

Computational artifacts can take many forms, such as programs, simulations, animations, visualizations, apps, or robotic systems, and serve as tools for innovation and problem-solving. Creating computational artifacts allows students to explore ideas, express creativity, and develop solutions to meaningful problems. Artifacts can be created by remixing existing components or designing entirely new ones. By designing and remixing existing components or creating entirely new solutions, students use computational artifacts to bring their ideas to life and address meaningful challenges.

Core Practice 6: Testing and refining computational artifacts

Testing and refinement are essential parts of developing high-quality computational artifacts. This iterative process involves identifying and correcting errors, comparing results to intended outcomes, and making adjustments to improve performance. The practice also includes consideration of user needs and feedback, and refining artifacts to enhance reliability, usability, and accessibility.

Core Practice 7: Communicating about computing

Communication is a key part of computer science, facilitating both personal expression and collaboration. Sharing ideas clearly with a variety of audiences, explaining how and why computation is used, and reflecting on the impact, better equips stakeholders for collaborative projects. Effective communication also entails writing meaningful comments, documenting work, and presenting thinking through different forms of media. Emphasis is placed on using precise language and tailoring communication to meet the needs of the intended audience.