Kansas Computer Science Model Standards
Grades 9-12

Concept	Subconcept	High School - All Students	Implementation Descriptions
Computing Systems	Devices	L1.CS.D.01 Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.	*Computing devices are often integrated with other systems, including biological, mechanical, and social systems. A medical device can be embedded inside a person to monitor and regulate his or her health, a hearing aid (a type of assistive device) can filter out certain frequencies and magnify others, a monitoring device installed in a motor vehicle can track a person's driving patterns and habits, and a facial recognition device can be integrated into a security system to identify a person. The creation of integrated or embedded systems is not an expectation at this level. Students might select an embedded device such as a car stereo, identify the types of data (radio station presets, volume level) and procedures (increase volume, store/recall saved station, mute) it includes, and explain how the implementation details are hidden from the user.*
	Hardware and Software	L1.CS.HS.01 Compare levels of abstraction and interactions between application software, system software, and hardware layers.	*At its most basic level, a computer is composed of physical hardware and electrical impulses. Multiple layers of software are built upon the hardware and interact with the layers above and below them to reduce complexity. System software manages a computing device's resources so that software can interact with hardware. For example, text editing software interacts with the operating system to receive input from the keyboard, convert the input to bits for storage, and interpret the bits as readable text to display on the monitor. System software is used on many different types of devices, such as smart TVs, assistive devices, virtual components, cloud components, and drones. For example, students may explore the progression from voltage to binary signal to logic gates to adders and so on. Knowledge of specific, advanced terms for computer architecture, such as BIOS, kernel, or bus, is not expected at this level.*
		L1.CS.HS.02 Compare computer systems and determine advantages and drawbacks of each system.	
	Input and Output	L1.CS.IO.01 Demonstrate efficeint use of input and output devices	*Fluency in educational and industry specific input and ouput devices including keyboard, mouse, touch screen, microphone, speakers, screen representations, printing, and other specific input and output devices.*
	Troubleshooting	L1.CS.T.01 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.	*Troubleshooting complex problems involves the use of multiple sources when researching, evaluating, and implementing potential solutions. Troubleshooting also relies on experience, such as when people recognize that a problem is similar to one they have seen before or adapt solutions that have worked in the past. Examples of complex troubleshooting strategies include resolving connectivity problems, adjusting system configurations and settings, ensuring hardware and software compatibility, and transferring data from one device to another. Students could create a flow chart, a job aid for a help desk employee, or an expert system.*
Networks & the Internet	Network Communication & Organization	L1.NI.NCO.01 Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing.	*Each device is assigned an address that uniquely identifies it on the network. Routers function by comparing IP addresses to determine the pathways packets should take to reach their destination. Switches function by comparing MAC addresses to determine which computers or network segments will receive frames. Students could use online network simulators to experiment with these factors.*
		L1.NI.NCO.02 Compare various security measures, considering tradeoffs between the usability and security of a computing system.	*Security measures may include physical security tokens, two-factor authentication, and biometric verification,but choosing secu*
	Cybersecurity	L1.NI.C.01 Recommend security measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.	*Security measures may include physical security tokens, two-factor authentication, and biometric verification. Potential security problems, such as denial-of-service attacks, ransomware, viruses, worms, spyware, and phishing, exemplify why sensitive data should be securely stored and transmitted. The timely and reliable access to data and information services by authorized users, referred to as availability, is ensured through adequate bandwidth, backups, and other measures. Students should systematically evaluate the feasibility of using computational tools to solve given problems or subproblems, such as through a cost-benefit analysis. Eventually, students should include more factors in their evaluations, such as how efficiency affects feasibility or whether a proposed approach raises ethical concerns.*
		L1.NI.C.02 Explain tradeoffs when selecting and implementing cybersecurity recommendations.	*Network security depends on a combination of hardware, software, and practices that control access to data and systems. The needs of users and the sensitivity of data determine the level of security implemented. Every security measure involves tradeoffs between the accessibility and security of the system. Students should be able to describe, justify, and document choices they make using terminology appropriate for the intended audience and purpose. Students could debate issues from the perspective of diverse audiences, including individuals, corporations, privacy advocates, security experts, and government.*
sis	Storage	L1.DA.S.01 Analyze storage types and locations.	
		L1.DA.S.02 Evaluate the tradeoffs in how data elements are organized and where data is stored.	*People make choices about how data elements are organized and where data is stored. These choices affect cost, speed, reliability, accessibility, privacy, and integrity. Students should evaluate whether a chosen solution is most appropriate for a particular problem. Students might consider the cost, speed, reliability, accessibility, privacy, and integrity tradeoffs between storing photo data on a mobile device versus in the cloud.*
	Collection	L1.DA.C.01 Collect and analyze data.	

Concept	Subconcept	High School - CS Career Focused Students	Implementation Descriptions
Computing Systems	Devices	L2.CS.D.02 Describe how internal and external parts of computing devices function to form a system.	
	Hardware and Software	L2.CS.HS.01 Categorize the roles of operating system software.	*Examples of roles could include memory management, data storage/retrieval, processes management, and access control.*
		L2.CS.HS.02 Compare options for building a computer systems and determine advantages and drawbacks of each piece and how it will affect the overall performance.	
	Input and Output	L2.CS.IO.01 Demonstrate use of course specific advanced input and output devices related to field	*Examples could include robotics, joysticks, motion sensors, movement sensors, gps, and various other cte specific course*
	Troubleshooting	L2.CS.T.01 Illustrate ways computing systems implement logic, input, and output through hardware components.	*Examples of components could include logic gates and IO pins.*
Networks & the Internet	Network Communication & Organization	L2.NI.NCO.01 Describe the issues that impact network functionality (e.g., bandwidth, load, delay, topology).	*Recommend use of free online network simulators to explore how these issues impact network functionality.*
		L2.NI.NCO.02 Give examples to illustrate how sensitive data can be affected by malware and other attacks.	
	Cybersecurity	L2.NI.C.01 Compare ways software developers protect devices and information from unauthorized access.	*Examples of security concerns to consider: encryption and authentication strategies, secure coding, and safeguarding keys.*
		L2.NI.C.02 Use encryption and decryption algorithms to transmit/ receive an encrypted message.	
sis	Storage	L2.DA.S.01 Translate and compare different bit representations of data types, such as characters, numbers, and images.	
		L2.DA.S.02 Analyze file systems created for keeping track of files on the hard disk.	
	Collection	L2.DA.C.01 Select data collection tools and techniques to generate data sets that support a claim or communicate information.	

Concept	Subconcept	High School - All Students	Implementation Descriptions
Data Analy	Visualization & Transformation	L1.DA.VT.01 Create interactive data visualizations using software tools to help others better understand real-world phenomena.	*People transform, generalize, simplify, and present large data sets in different ways to influence how other people interpret and understand the underlying information. Examples include visualization, aggregation, rearrangement, and application of mathematical operations. People use software tools or programming to create powerful, interactive data visualizations and perform a range of mathematical operations to transform and analyze data. Students should model phenomena as systems, with rules governing the interactions within the system and evaluate these models against real-world observations. For example, flocking behaviors, queueing, or life cycles. Google Fusion Tables can provide access to data visualization online.*
	Inference and Models	L1.DA.IM.01 Create computational models that represent the relationships among different elements of data collected from a phenomenon or process.	*Computational models make predictions about processes or phenomenon based on selected data and features. The amount, quality, and diversity of data and the features chosen can affect the quality of a model and ability to understand a system. Predictions or inferences are tested to validate models. Students should model phenomena as systems, with rules governing the interactions within the system. Students should analyze and evaluate these models against real-world observations. For example, students might create a simple producer–consumer ecosystem model using a programming tool. Eventually, they could progress to creating more complex and realistic interactions between species, such as predation, competition, or symbiosis, and evaluate the model based on data gathered from nature.*
Algorithms and Programming	Algorithms	L1.AP.A.01 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.	*A prototype is a computational artifact that demonstrates the core functionality of a product or process. Prototypes are useful for getting early feedback in the design process, and can yield insight into the feasibility of a product. The process of developing computational artifacts embraces both creative expression and the exploration of ideas to create prototypes and solve computational problems. Students create artifacts that are personally relevant or beneficial to their community and beyond. Students should develop artifacts in response to a task or a computational problem that demonstrate the performance, reusability, and ease of implementation of an algorithm.*
	Variables	L1.AP.V.01 Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.	*Students should be able to identify common features in multiple segments of code and substitute a single segment that uses lists (arrays) to account for the differences.*
	Control	L1.AP.C.01 Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.	*Implementation includes the choice of programming language, which affects the time and effort required to create a program. Readability refers to how clear the program is to other programmers and can be improved through documentation. The discussion of performance is limited to a theoretical understanding of execution time and storage requirements; a quantitative analysis is not expected. Control structures at this level may include conditional statements, loops, event handlers, and recursion. For example, students might compare the readability and program performance of iterative and recursive implementations of procedures that calculate the Fibonacci sequence.*
		L1.AP.C.02 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.	*In this context, relevant computational artifacts include programs, mobile apps, or web apps. Events can be user-initiated, such as a button press, or system-initiated, such as a timer firing. At previous levels, students have learned to create and call procedures. Here, students design procedures that are called by events. Students might create a mobile app that updates a list of nearby points of interest when the device detects that its location has been changed.*
		L1.AP.C.03 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.	*At this level, students should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist. For example, students could create an app to solve a community problem by connecting to an online database through an application programming interface (API)*
	Modularity	L1.AP.M.01 Create computational artifacts by systematically organizing, manipulating and/or processing data.	*Computational artifacts can be created by combining and modifying existing artifacts or by developing new artifacts. Examples of computational artifacts include programs, simulations, visualizations, digital animations, robotic systems, and apps. Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. Modules allow for better management of complex tasks. The focus at this level is understanding a program as a system with relationships between modules. The choice of implementation, such as programming language or paradigm, may vary. Students could incorporate computer vision libraries to increase the capabilities of a robot or leverage open-source JavaScript libraries to expand the functionality of a web application.*

Concept	Subconcept	High School - CS Career Focused Students	Implementation Descriptions
Data Analy	Visualization & Transformation	L2.DA.VT.01 Use data analysis tools and techniques to identify patterns in data representing complex systems.	*For example, identify trends in a dataset representing social media interactions, movie reviews, or shopping patterns.*
	Inference and Models	L2.DA.IM.01 Evaluate the ability of models and simulations to test and support the refinement of hypotheses. (e.g., flocking behaviors, life cycles, etc.)	
Algorithms and Programming	Algorithms	L2.AP.A.01 Describe how artificial intelligence algorithms drive many software and physical systems (e.g., digital advertising, autonomous robots, computer vision, pattern recognition, text analysis).	
		L2.AP.A.02 Describe how artificial intelligence drives many software and physical systems.	*Examples include digital ad delivery, self-driving cars, and credit card fraud detection.*
		L2.AP.A.03 Critically examine and trace classic algorithms (e.g., selection sort, insertion sort, binary search, linear search).	
		L2.AP.A.04 Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.	*Games do not have to be complex. Simple guessing games, Tic-Tac-Toe, or simple robot commands will be sufficient.*
		L2.AP.A.05 Use and adapt classic algorithms to solve computational problems.	*Examples could include sorting and searching.*
		L2.AP.A.06 Evaluate algorithms in terms of their efficiency, correctness, and clarity.	*Examples could include sorting and searching.*
	Variables	L2.AP.V.01 Compare and contrast simple data structures and their uses to simplify solutions, generalizing computational problems instead of repeatedly using primitive variables.	*Examples could include strings, lists, arrays, stacks, and queues.*
	Control	L2.AP.C.01 Trace the execution of repetition (e.g., loops, recursion), illustrating output and changes in values of named variables.	
	Modularity	L2.AP.M.01 Construct solutions to problems using student-created components, such as procedures, modules and/or objects.	

Concept	Subconcept	High School - All Students	Implementation Descriptions
		L1.AP.M.02 Systematically design and develop programs for broad audiences by incorporating feedback from users.	*Examples of programs could include games, utilities, and mobile applications. Students at lower levels collect feedback and revise programs. At this level, students should do so through a systematic process that includes feedback from broad audiences. Students might create a user satisfaction survey and brainstorm distribution methods that could yield feedback from a diverse audience, documenting the process they took to incorporate selected feedback in product revisions.*
	Program Development	L1.AP.PD.01 Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.	*Examples of software licenses include copyright, freeware, and the many open-source licensing schemes. At previous levels, students adhered to licensing schemes. At this level, they should consider licensing implications for their own work, especially when incorporating libraries and other resources. Students might consider two software libraries that address a similar need, justifying their choice based on the library that has the least restrictive license.*
		L1.AP.PD.02 Evaluate and refine computational artifacts to make them more usable and accessible.	*Testing and refinement is the deliberate and iterative process of improving a computational artifact. This process includes debugging (identifying and fixing errors) and comparing actual outcomes to intended outcomes. Students should respond to the changing needs and expectations of end users and improve the performance, reliability, usability, and accessibility of artifacts. For example, students could incorporate feedback from a variety of end users to help guide the size and placement of menus and buttons in a user interface.*
		L1.AP.PD.03 Design and develop computational artifacts working in team roles using collaborative tools.	*Collaborative tools could be as complex as source code version control system or as simple as a collaborative word processor. Team roles in pair programming are driver and navigator but could be more specialized in larger teams. As programs grow more complex, the choice of resources that aid program development becomes increasingly important and should be made by the students. Students might work as a team to develop a mobile application that addresses a problem relevant to the school or community, selecting appropriate tools to establish and manage the project timeline; design, share, and revise graphical user interface elements; and track planned, in-progress, and completed components.*
		L1.AP.PD.04 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.	*Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program; combinations of data and procedures; or independent, but interrelated, programs. The development of complex programs is aided by resources such as libraries and tools to edit and manage parts of the program.*
	Culture	L1.IC.C.01 Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.	*Computing may improve, harm, or maintain practices. Equity deficits, such as minimal exposure to computing, access to education, and training opportunities, are related to larger, systemic problems in society. Students should be able to evaluate the accessibility of a product to a broad group of end users, such as people who lack access to broadband or who have various disabilities. Students should also begin to identify potential bias during the design process to maximize accessibility in product design.*
		L1.IC.C.02 Test and refine computational artifacts to reduce bias and equity deficits.	*Biases could include incorrect assumptions developers have made about their user base. Equity deficits include minimal exposure to computing, access to education, and training opportunities. Students should begin to identify potential bias during the design process to maximize accessibility in product design and become aware of professionally accepted accessibility standards to evaluate computational artifacts for accessibility.*
		L1.IC.C.03 Demonstrate how a given algorithm applies to problems across disciplines.	*Computation can share features with disciplines such as art and music by algorithmically translating human intention into an artifact. Students should be able to identify real-world problems that span multiple disciplines, such as increasing bike safety with new helmet technology, and that can be solved computationally.*
	Social Interactions	L2.IC.SI.01 Compare and contrast the benefits and drawbacks of social media.	
mputing	History	L1.IC.H.01 Hypothesize the impact of the innovations of computing systems for the next decade.	

Concept	Subconcept	High School - CS Career Focused Students	Implementation Descriptions
		L2.AP.M.02 Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution.	*As students encounter complex, real-world problems that span multiple disciplines or social systems, they should decompose complex problems into manageable subproblems that could potentially be solved with programs or procedures that already exist. For example, students could create an app to solve a community problem by connecting to an online database through an application programming interface (API).*
		L2.AP.M.03 Demonstrate code reuse by creating programming solutions using libraries and APIs.	*Libraries and APIs can be student-created or common graphics libraries or maps APIs, for example.*
		L2.AP.PD.01 Plan and develop programs for broad audiences using a software life cycle process.	*Processes could include agile, spiral, or waterfall.*
		L2.AP.PD.02 Explain security issues that might lead to compromised computer programs.	*For example, common issues include lack of bounds checking, poor input validation, and circular references.*
	Program Development	L2.AP.PD.03 Develop programs for multiple computing platforms.	*Example platforms could include: computer desktop, web, or mobile.*
		L2.AP.PD.04 Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (code documentation) in a group software project.	*Group software projects can be assigned or student-selected.*
		L2.AP.PD.05 Develop and use a series of test cases to verify that a program performs according to its design specifications.	*At this level, students are expected to select their own test cases.*
		L2.AP.PD.06 Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).	*For instance, changes made to a method or function signature could break invocations of that method elsewhere in a system.*
		L2.AP.PD.07 Evaluate key qualities of a program through a process such as a code review.	*Examples of qualities could include correctness, usability, readability, efficiency, portability and scalability.*
		L2.AP.PD.08 Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems.	*Examples of features include blocks versus text, indentation versus curly braces, and high-level versus low- level.*
		L2.IC.C.01 Evaluate the beneficial and harmful effects that computational artifacts and innovations have on society.	
	Culture	L2.IC.C.02 Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society.	
		L2.IC.C.03 Design and implement a study that evaluates or predicts how computing has revolutionized an aspect of our culture and how it might evolve (e.g., education, healthcare, art/entertainment, energy).	*Areas to consider might include education, healthcare, art/entertainment, and energy.*
omputing	Social Interactions	L1.IC.SI.01 Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields.	
	History	L2.IC.H.01 Analyze trends of computing and how those trends have changed over time.	

Concept	Subconcept	High School - All Students	Implementation Descriptions
Impacts of C	Safety, Law, & Ethics	L1.IC.SLE.01 Explain the beneficial and harmful effects that intellectual property laws can have on innovation.	*Laws govern many aspects of computing, such as privacy, data, property, information, and identity. These laws can have beneficial and harmful effects, such as expediting or delaying advancements in computing and protecting or infringing upon people's rights. International differences in laws and ethics have implications for computing. For examples, laws that mandate the blocking of some file-sharing websites may reduce online piracy but can restrict the right to access information. Firewalls can be used to block harmful viruses and malware but can also be used for media censorship. Students should be aware of intellectual property laws and be able to explain how they are used to protect the interests of innovators and how patent trolls abuse the laws for financial gain.*
		L1.IC.SLE.02 Explain the privacy concerns related to the collection and generation of data through automated processes (e.g., how businesses, social media, and the government collects and uses data) that may not be evident to users.	*Data can be collected and aggregated across millions of people, even when they are not actively engaging with or physically near the data collection devices. This automated and nonevident collection can raise privacy concerns, such as social media sites mining an account even when the user is not online. Other examples include surveillance video used in a store to track customers for security or information about purchase habits or the monitoring of road traffic to change signals in real time to improve road efficiency without drivers being aware. Methods and devices for collecting data can differ by the amount of storage required, level of detail collected, and sampling rates.*
		L1.IC.SLE.03 Evaluate the social and economic implications of privacy in the context of safety, law, or ethics.	*Laws govern many aspects of computing, such as privacy, data, property, information, and identity. International differences in laws and ethics have implications for computing. Students might review case studies or current events which present an ethical dilemma when an individual's right to privacy is at odds with the safety, security, or wellbeing of a community.*
	Community Partnerships	L1.IC.CP.01 Explore computing, software, and data storage systems in local industries.	

Concept	Subconcept	High School - CS Career Focused Students	Implementation Descriptions
Impacts of C	Safety, Law, & Ethics	L2.IC.SLE.01 Debate laws and regulations that impact the development and use of software.	
		L2.IC.SLE.02 Determine ways to test the validity of information located online.	
		L2.IC.SLE.03 Evaluate the social and economic consequences of how law and ethics interact with digital aspects of privacy, data, property, information, and identity.	
	Community Partnerships	L2.IC.CP.01 Collaborate with local industry partners to design and implement a viable mentorship.	