Wisconsin Standards for Computer Science Alignment with CodeX Curriculum

Middle (Grades 6-8)	Unit 1	Unit 2	Unit 3
Algorithms and Programming			
AP1.a.6.m Decompose (break down) a computational problem into parts and create solutions for one or more parts.			
AP1.a.7.m Identify how sub-problems could be recombined to create something new (e.g., break down the individual parts that would be needed to program a certain type of game and then show how the parts could be reused in other types of games).			
AP2.a.6.m Develop programs, both independently and collaboratively, which include sequencing with nested loops and multiple branches [Clarification At this level, students may use block-based and/or text- based languages].	[1]		
AP2.a.7.m Produce computational artifacts with broad accessibility and usability through careful consideration of diverse needs and wants of the community.	[2]		
AP2.a.8.m Use an iterative design process (e.g., define the problem; generate ideas; build, test, and improve solutions) to solve computational problems, both independently and collaboratively.			
AP2.a.9.m Create variables that represent different types of data and manipulate their values.	[3]		
AP3.a.3.m Provide proper attribution when code is borrowed or built upon.			
AP3.b.5.m Discuss how algorithms have impacted society— both the beneficial and harmful effects.			
AP3.b.6.m 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].			
AP3.b.7.m Modify existing code to change its functionality and discuss the variety of ways in which to do this.	[4]		
AP3.c.1.m 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].	[5]		
AP3.c.2.m Use documentation regarding code to modify programs.	[6]		
AP4.a.3.m Define and use functions/ 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, functions or procedures with parameters].	[7]		
AP5.a.5.m Solicit and integrate peer feedback as appropriate to develop or refine a program.			
AP5.b.2.m Analyze team members' strengths and use them to foster an inclusive computing culture.			
AP6.a.3.m Use testing and debugging methods to ensure program correctness and completeness.	[8]		
AP6.b.2.m Apply a rubric to determine if and how well a program meets objectives.			
Computing Systems			
CS1.a.5.m Justify the suitability of 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).			
CS2.a.3.m 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).			
CS3.a.1.m Analyze the relationship between a device's computational components and its capabilities. (e.g., computing systems include not only computers, but also cars, microwaves, smartphones, traffic lights, and flash drives).			
CS4.a.1.m Extend or modify existing programs to add simple 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).	[9]		
Data and Analysis			
DA1.a.3.m Represent data using different encoding schemes (e.g., binary, Unicode, Morse code, shorthand, student-created codes).			
DA2.a.3.m Gather and organize multiple quantitative data elements using a computational tool (e.g., spreadsheet software).			
DA2.b.3.m Develop a strategy to 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.			
DA3.a.4.m Describe how different formats of stored data represent tradeoffs between quality and size. [Clarification compare examples of music, text and/or image formats].			
DA3.a.5.m 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).			
DA4.a.4.m Revise computational models to more accurately reflect real-world systems (e.g., ecosystems, epidemics, spread of ideas).			
DA4.a.5.m Modify an existing computational model to emphasize key features and relationships within a system. (A model can be used to simulate events, examine theories and inferences, or make predictions).			
Impacts of Computing			
IC1.a.4.m Provide examples of how computational artifacts and devices impact health and wellbeing, both positively and negatively, locally and globally (e.g., effects of globalization, and automation).			
IC1.a.5.m Explain how computer science fosters innovation and can enhance careers and disciplines.			
IC1.b.3.m Analyze and present beneficial and harmful effects of personal electronic communication and social electronic communication.			
IC1.b.4.m Describe ways in which the internet impacts global communication and collaborating.			
IC2.a.2.m Explain the impact of the digital divide (i.e., uneven access to computing, computing education, and interfaces) on access to critical information.			
IC2.b.2.m Critically evaluate and redesign a computational artifact to remove barriers to universal access (e.g., using captions on images, high contrast colors, and/or larger font sizes).			
IC2.c.4.m Use the internet ethically and safely to work with a group of people who are not physically near to solve a problem or reach a goal.			
IC3.a.2.m Understand laws associated with digital information (e.g., intellectual property, fair use, and Creative Commons).			
IC3.a.3.m 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).			
IC3.b.4.m Analyze and summarize negative and positive impacts of using data and information to categorize people, predict behavior, and make recommenda- tions 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).			
Networks & the Internet			
NI1.a.4.m Analyze and summarize security risks associated with weak passwords, lack of encryption, insecure transactions, and persistence of data.			
NI1.a.5.m Understand security issues with general computer use			
NI1.b.2.m Explain the principles of information security (confidentiality, integrity, availability) and authentication techniques.			
NI2.a.6.m Simulate how information is transmitted as packets through multiple devices over the internet and networks.			
NI2.a.7.m Explain, using basic terms, how a wireless or cellular network allows internet information to be transmitted from a server to a user device.			
NI2.b.2.m Define the term protocol, provide an example of protocols in daily life, and explain their use on the internet.			
NI2.c.3.m Explain the hierarchical structure of the Internet Domain Name System (IDNS).			
NI2.d.2.m Encode and decode text- based messages using basic algorithms (e.g., shift cipher, substitution cipher).			

Wisconsin Standards for Computer Science Alignment with CodeX Curriculum

High (Grades 9-12)	Unit 1	Unit 2	Unit 3
Algorithms and Programming			
AP1.a.8.h Analyze a problem and design and implement an algorithmic solution using sequence, selection, and iteration.	[10]		
AP1.a.9.h Explain and demonstrate how modeling and simulation can be used to explore natural phenomena (e.g., flocking behaviors, queueing, life cycles).			
AP2.a.10.h Use user-centered research and design techniques (e.g., surveys, interviews) to create software solutions.			
AP2.a.11.h Integrate grade-level appropriate mathematical techniques, concepts, and processes in the creation of computational artifacts.			
AP2.a.12.h 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).	[11]		
AP3.a.4.h Compare and contrast various software licensing schemes (e.g., open source, freeware, commercial).			
AP3.b.8.h Evaluate and analyze how algorithms have impacted our society and discuss the benefits and harmful impacts of a variety of technological innovations.			
AP3.c.4.h Write appropriate documentation for programs.	[12]		
AP3.c.6.h Use online resources to answer technical questions.			
AP4.a.4.h Demonstrate the value of abstraction for managing problem complexity (e.g., using a list instead of discrete variables).		[13]	
AP4.a.5.h Understand the notion of hierarchy and abstraction in high-level languages, translation, instruction sets, and logic circuits.			
AP4.a.6.h Deconstruct a complex problem into simpler parts using predefined constructs (e.g., functions and parameters and/or classes).	[14]		
AP5.a.6.h Design and develop a software artifact working in a team.			
AP5.a.7.h 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).			
AP5.b.3.h Create design teams taking into account the strengths and perspectives of potential team members.			
AP6.a.4.h Use a systematic approach and debugging tools to independently debug a program (e.g., setting breakpoints, inspecting variables with a debugger).	[15]		
Computing Systems			
CS1.a.6.h 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).			
CS2.a.4.h Devise a systematic process to identify the source of a problem within individual and connected devices (e.g., research, investigate, problem solve).			
CS3.a.2.h 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.			
CS4.a.2.h 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).	[16]		
Data and Analysis			
DA1.a.4.h Convert between binary, decimal, and hexadecimal representations of data (e.g., convert hexadecimal color codes to decimal percentages, ASCII/ Unicode representation).			
DA1.a.5.h Analyze the representation tradeoffs among various forms of digital information (e.g., lossy vs. lossless compression, encrypted vs. unencrypted, various image representations).			
DA2.a.4.h Discuss techniques used to store, process, and retrieve different amounts of information (e.g., files, databases, data warehouses).			
DA2.b.4.h 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).			
DA3.a.6.h Use computational tools to collect, transform, and organize data about a problem to explain to others.			
DA4.a.6.h Create computational models that simulate real- world systems (e.g., ecosystems, epidemics, spread of ideas).			
Impacts of Computing			
IC1.a.6.h Debate the social and economic implications associated with ethical and unethical computing practices (e.g., intellectual property rights, hacktivism, software piracy, new computers shipped with malware).			
IC1.a.7.h 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).			
IC1.a.8.h Compare and debate the positive and negative impacts of computing on behavior and culture (e.g., evolution from hitchhiking to ride-sharing apps, online accommodation rental services).			
IC1.a.9.h Describe how computation shares features with art and music by translating human intention into an artifact.			
IC1.b.5.h Evaluate the negative impacts of electronic communication on personal relationships and evaluate differences between face- to-face and electronic communication.			
IC2.b.3.h Design a user interface (e.g., web pages, mobile applications, animations) to be more inclusive and accessible, minimizing the impact of the designer's inherent bias			
IC2.c.5.h Ethically and safely 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 platform, or contribute an online article).			
IC2.c.6.h Demonstrate how computing enables new forms of experience, expression, communication, and collaboration.			
IC3.a.4.h Compare and contrast information access and distribution rights.			
IC3.b.5.h Research and understand misuses of private digital information in our society.			
IC3.b.6.h Debate laws regarding an individual's digital privacy and be able to explain the main arguments from multiple perspectives.			
Networks & the Internet			
NI1.a.6.h Provide examples of personal data that should be kept secure and the methods by which individuals keep their private data secure.			
NI1.b.3.h Compare and contrast multiple viewpoints on cybersecurity (e.g., from the perspective of security experts, privacy advocates, national security).			
NI1.b.4.h Identify digital and physical strategies to secure networks and discuss the tradeoffs between ease of access and need for security.			
NI2.a.8.h 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).			
NI2.b.3.h Describe key protocols and underlying processes of internet-based services (e.g., http/https and Simple Mail Transfer Protocol (SMTP) or Internet Message Access Protocol (IMAP), routing protocols).			
NI2.d.3.h Write a program that performs basic encryption (e.g., shift cipher, substitution cipher).			

Wisconsin Standards for Computer Science Alignment with CodeX Curriculum

Performance indicators marked with a (+) for grades 9-12 represent advanced CS learning expectations for students with aspirations toward careers and postsecondary studies in computing disciplines.	Unit 1	Unit 2	Unit 3
Algorithms and Programming			
AP1.a.10.h (+) Provide examples of computationally solvable problems and difficult-to- solve problems.			
AP1.a.11.h (+) Decompose a large-scale computational problem by identifying generalizable patterns and applying them in a solution.			
AP1.a.12.h (+) Illustrate the flow of execution of a recursive algorithm.	[17]		
AP1.a.13.h (+) Describe how parallel processing can be used to solve large computational problems (e.g., SETI at Home, FoldIt).			
AP1.a.14.h (+) Develop and use a series of test cases to verify that a program performs according to its design specifications.			
AP1.a.15.h (+) Explain the value of heuristic algorithms (discovery methods) to approximate solutions for difficult-to-solve computational problems.			
AP2.a.13.h (+) Decompose a computational problem by creating new data types, functions, or classes.		[18]	
AP2.a.14.h (+) Develop programs for multiple computing platforms (e.g., computer desktop, web, mobile).			
AP2.a.15.h (+) Implement an Artificial Intelligence (AI) algorithm to play a game against a human opponent or solve a problem.			
AP2.a.16.h (+) Demonstrate code reuse by creating programming solutions using libraries and application program interfaces (APIs) (e.g., graphics libraries, maps, API).	[19]		
AP3.b.9.h (+) 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).			
AP3.b.10.h (+) Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).	[20]		
AP3.c.3.h (+) Describe how Artificial Intelligence (AI) drives many software and physical systems (e.g., autonomous robots, computer vision, pattern recognition, text analysis).			
AP3.c.5.h (+) Use application programming interface (APIs) documentation resources.			
AP4.a.7.h (+) Compare and contrast fundamental data structures and their uses (e.g., lists, maps, arrays, stacks, queues, trees, graphs).			
AP4.a.8.h (+) Critically analyze and evaluate classic algorithms (e.g., sorting, searching) and use in different contexts, adapting as appropriate.			
AP4.a.9.h (+) 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).			
AP4.a.10.h (+) Define the functionality of an abstraction without implementing the abstraction.			
AP4.a.11.h (+) Evaluate algorithms (e.g., sorting, searching) in terms of their efficiency, correctness, and clarity.			
AP4.a.12.h (+) Identify programming language features that can be used to define or specify an abstraction.			
AP4.a.13.h (+) Identify abstractions used in a solution (program or software artifact) and reuse those abstractions to solve a different problem.			
AP5.a.8.h (+) Demonstrate software life cycle processes (e.g., spiral, waterfall) by participating on software project teams (e.g., community service project with real-world clients).			
AP5.a.9.h (+) Use version control systems, integrated development environments (IDEs), and collaboration tools and practices (code documentation) in a group software project.			
AP6.b.3.h (+) Evaluate key qualities of a program (e.g., correctness, usability, readability, efficiency, portability, scalability) through a process such as a code review.	[21]		
Computing Systems			
CS1.a.7.h (+) Identify the functionality of various categories of hardware components and communication between them (e.g., physical layers, logic gates, chips, input and output devices).			
CS1.b.3.h (+) Explain the role of operating systems (e.g., how programs are stored in memory, how data is organized and retrieved, how processes are managed and multi-tasked).			
CS3.a.3.h (+) Describe the steps necessary for a computer to execute high-level source code (e.g., compilation to machine language, interpretation, fetch-decode-execute cycle).			
CS4.a.3.h (+) Create a new artifact that uses a variety of forms of inputs and outputs (e.g., inputs such as sensors, mouse clicks, data sets; outputs such as text, graphics, sounds).	[22]		
Data and Analysis			
DA1.a.6.h (+) Discuss how data sequences (e.g., binary, hexadecimal, octal) can be interpreted in a variety of forms (e.g., instructions, numbers, text, sound, image).			
DA2.a.5.h (+) Use various data collection techniques for different types of computational problems (e.g., mobile device Global Positioning System (GPS), user surveys, embedded system sensors, open data sets, social media data sets).			
DA4.a.7.h (+) Evaluate the ability of models and simulations to formulate, refine, and test hypotheses.			
DA4.b.1.h (+) Use data analysis to identify significant patterns in complex systems (e.g., take existing data sets and make sense of them).			
DA4.b.2.h (+) Identify mathematical and computational patterns through modeling and simulation (e.g., regression, queueing theory, discrete event simulation)			
Impacts of Computing			
IC1.a.10.h (+) Develop criteria to evaluate the beneficial and harmful effects of computing innovations on people and society.			
IC1.b.6.h (+) Create a list of practices that individuals and organizations can use to encourage proper use of both electronic and face-to- face communication.			
IC1.b.7.h (+) Evaluate the negative impacts on societal discourse caused by social media and electronic communities.			
IC2.a.3.h (+) Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society.			
IC3.c.1.h (+) Design and implement a study that evaluates how computation has revolutionized an aspect of our culture or predicts how an aspect might evolve (e.g., education, healthcare, art/entertainment, energy).			
IC3.c.2.h (+) Debate laws and regulations that impact the development and use of software and be able to explain the main arguments from multiple perspectives.			
Networks & the Internet			
NI1.a.7.h (+) 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).			
NI2.a.9.h (+) Explain ways in which the internet is decentralized and fault- tolerant.			
NI2.a.10.h (+) Simulate and discuss the issues (e.g., bandwidth, load, delay, topology) that impact network functionality (e.g., use free network simulators).			
NI2.c.4.h (+) Evaluate how the hierarchical nature of the Domain Name System helps the internet work efficiently.			
NI2.d.4.h (+) Explain the features of public key cryptography.			
NI2.d.5.h (+) Explore security policies by implementing and comparing encryption and authentication strategies (e.g., secure coding, safeguarding keys).			

[1] Mission 6 introduces the use of nested loops
Mission 9 uses multiple branching

[2] This can be done with remixes depending on the teacher's rubric

[3] 3.8 begins the use of variables
5.5 discusses descriptive variable naming

[4] These are the remixes that are introduced in Mission 4

[5] Flowcharts and pseudocodes are introduced in the teachers' manual

[6] 5.5 introduces the use of comments

[7] Mission 4 begins the use of functions

[8] 3.5 introduces the debugger

[9] These are the remixes beginning in Mission 4

[10] This begins in Mission 4

[11] Many of our missions utilize and explain the use of sensors

[12] 5.5 introduces the use of comments

[13] 7.5 begins the use of lists

[14] Mission 4 begins the use of functions

[15] 3.5 introduces the debugger

[16] Writing remixes that utilize the sensors accomplishes this

[17] Flowcharts are introduced in the teachers' manual

[18] 9.3 introduces creating your own functions

[19] All missions use libraries and when new ones are introduced they are explained

[20] These are the remixes depending on the rubric the teachers gives

[21] Code Tracing Charts are introduced in the teachers' manual

[22] Most of our missions do this