Massachusetts Standards Alignment with CodeX Curriculum

Computational Thinking	Unit 1	Unit 2	Unit 3
3-5.CT.a Abstraction			
3-5.CT.a.1 Use numbers or letters to represent information in another form (e.g., secret codes, Roman numerals, abbreviations).			
3-5.CT.a.2 Organize information in different ways to make it more useful/relevant (e.g., sorting, tables).			
3-5.CT.a.3 Make a list of sub-problems to consider, while addressing a larger problem.	[1]		
3-5.CT.b Algorithms			
3-5.CT.b.1 Define an algorithm as a sequence of instructions that can be processed by a computer.			
3-5.CT.b.2 Recognize that different solutions exist for the same problem (or sub-problem).			
3-5.CT.b.3 Use logical reasoning to predict outcomes of an algorithm.			
3-5.CT.b.4 Individually and collaboratively create an algorithm to solve a problem (e.g., move a character/robot/person through a maze).			
3-5.CT.b.5 Detect and correct logical errors in various algorithms (e.g., written, mapped, live action, or digital).			
3-5.CT.c Data			
3-5.CT.c.1 Describe examples of databases from everyday life (e.g., library catalogs, school records, telephone directories, contact lists).			
3-5.CT.c.2 Collect and manipulate data to answer a question using a variety of computing methods (e.g., sorting, totaling, averaging) and tools (such as a spreadsheet) to collect, organize, graph, and analyze data.			
3-5.CT.d Programming and Development			
3-5.CT.d.1 Individually and collaboratively create, test, and modify a program in a graphical environment (e.g., block-based visual programming language).			
3-5.CT.d.2 Use arithmetic operators, conditionals, and repetition in programs.	[2]		
3-5.CT.d.3 Use interactive debugging to detect and correct simple program errors.	[3]		
3-5.CT.d.4 Recognize that programs need known starting values (e.g., set initial score to zero in a game).	[4]		
3-5.CT.e Modeling and Simulation			
3-5.CT.e.1 Individually and collaboratively create a simple model of a system (e.g., water cycle, solar system) and explain what the model shows and does not show			
3-5.CT.e.2 Identify the concepts, features, and behaviors illustrated by a simulation (e.g., object motion, weather, ecosystem, predator/prey) and those that were not included.			
3-5.CT.e.3 Individually and collaboratively use data from a simulation to answer a question.			

Massachusetts Standards Alignment with CodeX Curriculum

Computational Thinking	Unit 1	Unit 2	Unit 3
6-8.CT.a Abstraction			
6-8.CT.a.1 Describe how data is abstracted by listing attributes of everyday items to represent, order and compare those items (e.g., street address as an abstraction for locations; car make, model, and license plate number as an abstraction for cars).			
6-8.CT.a.2 Define a simple function that represents a more complex task/problem and can be reused to solve similar tasks/problems.		[5]	
6-8.CT.a.3 Use decomposition to define and apply a hierarchical classification scheme to a complex system, such as the human body, animal classification, or in computing.			
6-8.CT.b Algorithms			
6-8.CT.b.1 Design solutions that use repetition and conditionals.	[6]		
6-8.CT.b.2 Use logical reasoning to predict outputs given varying inputs.			
6-8.CT.b.3 Individually and collaboratively decompose a problem and create a sub-solution for each of its parts (e.g., video game, robot obstacle course, making dinner).			
6-8.CT.b.4 Recognize that more than one algorithm can solve a given problem.			
6-8.CT.b.5 Recognize that boundaries need to be taken into account for an algorithm to produce correct results.			
6-8.CT.c Data			
6-8.CT.c.1 Demonstrate that numbers can be represented in different base systems (e.g., binary, octal, and hexadecimal) and text can be represented in different ways (e.g., American Standard Code for Information Interchange [ASCII]).			
6-8.CT.c.2 Describe how computers store, manipulate, and transfer data types and files (e.g., integers, real numbers, Boolean Operators) in a binary system.			
6-8.CT.c.3 Create, modify, and use a database (e.g., define field formats, add new records, manipulate data), individually and collaboratively, to analyze data and propose solutions for a task/problem.			
6-8.CT.c.4 Perform a variety of operations such as sorting, filtering, and searching in a database to organize and display information in a variety of ways such as number formats (scientific notation and percentages), charts, tables, and graphs.			
6-8.CT.c.5 Select and use data-collection technology (e.g., probes, handheld devices, geographic mapping systems) to individually and collaboratively gather, view, organize, analyze, and report results for content-related problems.			
6-8.CT.d Programming and Development			
6-8.CT.d.1 Individually and collaboratively compare algorithms to solve a problem, based on a given criteria (e.g., time, resource, accessibility).			
6-8.CT.d.2 Use functions to hide the detail in a program.	[7]		
6-8.CT.d.3 Create a program, individually and collaboratively, that implements an algorithm to achieve a given goal.			
6-8.CT.d.4 Implement problem solutions using a programming language, including all of the following: looping behavior, conditional statements, expressions, variables, and functions.	[8]		
6-8.CT.d.5 Trace programs step-by-step in order to predict their behavior.	[9]		
6-8.CT.d.6 Use an iterative approach to development and debugging to understand the dimensions of a problem clearly.			
6-8.CT.e Modeling and Simulation			
6-8.CT.e.1 Create a model of a real-world system and explain why some details, features and behaviors were required in the model and why some could be ignored.			
6-8.CT.e.2 Use and modify simulations to analyze and illustrate a concept in depth (e.g., light rays/mechanical waves interaction with materials, genetic variation).			
6-8.CT.e.3 Select and use computer simulations, individually and collaboratively, to gather, view, analyze, and report results for content-related problems (e.g., migration, trade, cellular function).			

Massachusetts Standards Alignment with CodeX Curriculum

	Unit 1	Unit 2	Unit 3
9-12.CS.a Computing Devices			
9-12.CS.a.1 Select computing devices (e.g., probe, sensor, tablet) to accomplish a real-world task (e.g., collecting data in a field experiment) and justify the selection.			
9-12.CS.a.2 Examine how the components of computing devices are controlled by and react to programmed commands.	[10]		
9-12.CS.a.3 Apply strategies for identifying and solving routine hardware and software problems that occur in everyday life (e.g., update software patches, virus scan, empty trash, run utility software, close all programs, reboot, use help sources).			
9-12.CS.a.4 Explain and demonstrate how specialized computing devices can be used for problem solving, decision-making and creativity in all subject areas.			
9-12.CS.a.5 Describe how computing devices manage and allocate shared resources (e.g., memory, Central Processing Unit [CPU])			
9-12.CS.a.6 Examine the historical rate of change in computing devices (e.g., power/energy, computation capacity, speed, size, ease of use) and discuss the implications for the future.			
9-12.CS.b Human and Computer Partnerships			
9-12.CS.b.1 Identify a problem that cannot be solved by humans or machines alone and design a solution for it by decomposing the task into sub-problems suited for a human or machine to accomplish (e.g., a human-computer team playing chess, forecasting weather, piloting airplanes).			
9-12.CS.c Networks			
9-12.CS.c.1 Explain how network topologies and protocols enable users, devices, and systems to communicate with each other.			
9-12.CS.c.2 Examine common network vulnerabilities (e.g., cyberattacks, identity theft, privacy) and their associated responses.			
9-12.CS.c.3 Examine the issues (e.g., latency, bandwidth, firewalls, server capability) that impact network functionality.			
9-12.CS.d Services			
9-12.CS.d.1 Compare the value of using an existing service versus building the equivalent functionality (e.g., using a reference search engine versus creating a database of references for a project).			
9-12.CS.d.2 Explain the concept of quality of service (e.g., security, availability, performance) for services providers (e.g., online storefronts that must supply secure transactions for buyer and seller).			
9-12.CT.a Abstraction			
9-12.CT.a.1 Discuss and give an example of the value of generalizing and decomposing aspects of a problem in order to solve it more effectively			
9-12.CT.b Algorithms			
9-12.CT.b.1 Recognize that the design of an algorithm is distinct from its expression in a programming language.			
9-12.CT.b.2 Represent algorithms using structured language, such as pseudocode.			
9-12.CT.b.3 Explain how a recursive solution to a problem repeatedly applies the same solution to smaller instances of the problem.			
9-12.CT.b.4 Describe that there are ways to characterize how well algorithms perform and that two algorithms can perform differently for the same task.			
9-12.CT.b.5 Explain that there are some problems which cannot be computationally solved.			
9-12.CT.c Data			
9-12.CT.c.1 Describe how data types, structures, and compression in programs affect data storage and quality (e.g., digital image file sizes are affected by resolution and color depth).			
9-12.CT.c.2 Create an appropriate multidimensional data structure that can be filtered, sorted, and searched (e.g., array, list, record).			
9-12.CT.c.3 Create, evaluate, and revise data visualization for communication and knowledge			
9-12.CT.c.4 Analyze a complex data set to answer a question or test a hypothesis (e.g., analyze a large set of weather or financial data to predict future patterns).			
9-12.CT.c.5 Identify different problems (e.g., large or multipart problems, problems that need specific expertise, problems that affect many constituents) that can benefit from collaboration when processing and analyzing data to develop new insights and knowledge.			
9-12.CT.d Programming and Development			
9-12.CT.d.1 Use a development process in creating a computational artifact that leads to a minimum viable product and includes reflection, analysis, and iteration (e.g., a data-set analysis program for a science and engineering fair, capstone project that includes a program, term research project based on program data).	[11]		
9-12.CT.d.2 Decompose a problem by defining functions which accept parameters and produce return values.			
9-12.CT.d.3 Select the appropriate data structure to represent information for a given problem (e.g., records, arrays, lists).			
9-12.CT.d.4 Analyze trade-offs among multiple approaches to solve a given problem (e.g., space/time performance, maintainability, correctness, elegance).			
9-12.CT.d.5 Use appropriate looping structures in programs (e.g., FOR, WHILE, RECURSION).	[12]		
9-12.CT.d.6 Use appropriate conditional structures in programs (e.g., IF-THEN, IF-THEN-ELSE, SWITCH).	[13]		
9-12.CT.d.7 Use a programming language or tool feature correctly to enforce operator precedence.			
9-12.CT.d.8 Use global and local scope appropriately in program design (e.g., for variables).	[14]		
9-12.CT.d.9 Select and employ an appropriate component or library to facilitate programming 2016 Massachusetts Digital Literacy and Computer Science Curriculum Framework 35 solutions (e.g., turtle, Global Positioning System [GPS], statistics library).			
9-12.CT.d.10 Use an iterative design process, including learning from making mistakes, to gain a better understanding of the problem domain.			
9-12.CT.d.11 Engage in systematic testing and debugging methods to ensure program correctness	[15]		
9-12.CT.d.12 Demonstrate how to document a program so that others can understand its design and implementation	[16]		
9-12.CT.e Modeling and Simulation			
9-12.CT.e.1 Create models and simulations to help formulate, test, and refine hypotheses.			
9-12.CT.e.2 Form a model from a hypothesis generated from research and run a simulation to collect and analyze data to test that hypothesis.			

[1] This could be covered in the pseudocodes and Flowcharts of remixes depending on teacher requirements

[2] These begin in Mission 4

[3] 3.5 introduces the debugger

[4] Mission 3 introduces the use of variables
Mission 6 begins variable manipulation while running the code

[5] Mission 9 begins the creation of your own function

[6] This begins in Mission 4

[7] Mission 4 begins the use of functions

[8] These all are introduced in Mission 4

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

[10] All of our missions are programmed commands controlling computing devices

[11] These could be the remixes depending on the rubric the teacher gives

[12] Mission 4 begins the use of loops

[13] These are introduced in Mission 4

[14] 3.8 begins the use of variables

[15] 3.5 introduces the debugger

[16] 5.5 introduces comments