Tennessee CS Standards Alignment with CodeX Curriculum

	Unit 1	Unit 2	Unit 3
Coding and Computer Programming			
5.CCP.1 Identify and describe the role of various input and output devices and components that are within an interdependent system with a common purpose.			
5.CCP.2 Investigate and trace a bundle of information through a series of packets and different systems via a protocol.			
5.CCP.3 Decompose (break down) complex real-world problems in multiple ways that use variables to develop a solution or procedure based on data.	[1]		
5.CCP.4 Create an algorithm which includes control structures to solve a problem using visual block-based and/or textbased programming language both collaboratively and individually.	[2]		
5.CCP.5 Decompose complex code into subsections or subprograms for reuse into other programs.	[3]		
5.CCP.6 Decompose a piece of code with the intent to debug a section of code	[4]		
5.CCP.7 Formulate alternative uses for software and hardware for various members of society.			

Tennessee CS Standards Alignment with CodeX Curriculum			
	Unit 1	Unit 2	Unit 3
Coding and Computer Programming			
CCP.1 Identify the advantages, disadvantages and unintended consequences of computing devices.			
CCP.2 Analyze the relationship between human and computer interactions to improve the device. For example, student A watches student B use a simple communication device. Student A updates the tool for improved use.			
CCP.3 Identify and describe multiple considerations and tradeoffs when designing or selecting computing system, such as functionality, cost, size, speed, accessibility, and aesthetics.			
CCP.4 Construct optimized models of computing systems.			
CCP.5 Create structured processes to troubleshoot problems with computing systems.	[5]		
CCP.6 Define protocols in relation to a set of rules.			
CCP.7 Construct protocols that can be used to share information between people or devices. For example, a binary communication protocol using lights.			
CCP.8 Compare the relative strengths and weaknesses of unique protocols considering security, speed, and reliability.			
CCP.9 Create models of networks that include packets and domain name server (DNS).			
CCP.10 Identify steps to ensure security measures are in place to safeguard online information			
CCP.11 Create cyphers to encrypt data that can be transferred between users.			
CCP.12 Explain how encryption can be used to safeguard data that is sent across a network.			
CCP.13 Evaluate the accuracy and precision of various forms of data collection.			
CCP.14 Identify and define the limiting factors to specific forms of data collection.			
CCP.15 Describe how different formats of stored data represent tradeoffs between quality and size.			
CCP.16 Represent data using different encoding schemes, such as binary, Unicode, Morse code, shorthand, student-created codes.			
CCP.17 Explain the processes used to collect, transform, and analyze data to solve a problem using computational tools.			
CCP.18 Revise variables and constants in computational models to more accurately reflect real-world systems. For example in an ecosystem model, introducing predators as a new variable.			
CCP.19 Solicit and integrate peer feedback as appropriate to develop or refine a program.			
CCP.20 Compare different algorithms that may be used to solve the same problem in terms of their speed, clarity, and size.			
CCP.21 Provide proper attribution when code is borrowed or built upon.			
CCP.22 Interpret the flow of execution of algorithms and predict their outcomes.	[6]		
CCP.23 Design, develop, and present computational artifacts such as mobile applications that address social problems both independently and collaboratively.			
CCP.24 Develop programs, both independently and collaboratively, that include sequences with nested loops and multiple branches. (Clarification: At this level, students may use block- based and/or text-based programming languages.)	[7]		

Tennessee CS Standards Alignment with CodeX Curriculum			
	Unit 1	Unit 2	Unit 3
CCP.25 Identify the purpose of variables in relation to programming.	[8]		
CCP.26 Create variables that represent different types of data and manipulate their values.	[9]		
CCP.27 Define and use procedures that hide the complexity of a task and can be reused to solve similar tasks. (Clarification: Students use and modify, but do not necessarily create, procedures with parameters.)			
CCP.28 Decompose a problem into parts and create solutions for each part.			
CCP.29 Use an iterative design process (e.g., define the problem, generate ideas, build, test, and improve solutions) to solve problems, both independently and collaboratively.	[10]		
CCP.30 Analyze the positive and negative impacts of computing technology.			
CCP.31 Recognize there are tradeoffs in computing.			
CCP.32 Explain how social interactions can allow for multiple viewpoints.			
CCP.33 Demonstrate an understanding of digital security.			

Tennessee CS Standards Alignment with CodeX Curriculum			
Course Code: C10H14	Unit 1	Unit 2	Unit 3
Computer Programming Overview			
1) Using news articles and instructional materials, investigate key milestones in the development of computers and logical devises. Create and present a document and/or illustration depicting the timeline of development that led to modern-day operating systems, programmable controllers, and widespread digital communications via the Internet and wireless networks, citing specific textual evidence.			
2) Compare and contrast the benefits, features, and typical applications of common modern programming languages and environments. Craft an argument to defend the choice of a certain language to solve a particular problem, developing claim(s) and counterclaim(s) with specific textual evidence and reasoning.			
Ethics			
3) Using news articles and text of legislation, analyze ethical programming practices, including but not limited to the issues of confidentiality, privacy, piracy, fraud and misuse, liability, copyright, open source software, trade secrets, and sabotage. For example, research and report on the effects of unethical programming practices on a business.			
Programming Skills			
4) Differentiate between system-level and application solutions, and identify an appropriate code-based strategy to solve a given problem. For example, given a file management problem, determine when a command-line script will be more efficient than a high-level program solution.			
5a) Apply the system management tools present in a programming development environment to select the most appropriate programming language for the task at hand.			
5b) Apply the system management tools present in a programming development environment to develop syntactically correct program code using current best practices and emerging classes of development techniques.	[11]		
5c) Apply the system management tools present in a programming development environment to use a compiler to interpret the source code and produce executable program code.			
6a) In the process of developing and implementing programming solutions, develop strategies that work within the constraints of major operating system fundamentals, such as security protocols and procedures for accessing files and folders.			
6b) In the process of developing and implementing programming solutions, develop strategies that work within the constraints of major operating system fundamentals, such as file management syntax requirements, including but not limited to creating, naming, organizing, copying, moving, and deleting files.			
6c) In the process of developing and implementing programming solutions, develop strategies that work within the constraints of major operating system fundamentals, such as file naming conventions, as they apply across multiple software applications and file types.			
7) Write pseudocode and construct a flowchart for a process before starting to develop the program code. For example, code and flowchart a simple process that takes an integer and report whether it is odd or even.	[12]		
8a) Organize and develop a plan to acquire and manage the data values for a process, including the following: Data types, such as string, numeric, character, integer, and date.			
8b) Organize and develop a plan to acquire and manage the data values for a process, including the following: Program variable names	[13]		
8c) Organize and develop a plan to acquire and manage the data values for a process, including the following: Variables and constants	[14]		
8d) Organize and develop a plan to acquire and manage the data values for a process, including the following: Arrays (at least one- and two-dimensional), subscripts			
8e) Organize and develop a plan to acquire and manage the data values for a process, including the following: Input from files and user responses			
8f) Organize and develop a plan to acquire and manage the data values for a process, including the following: Output to files and reports			
9) Using a programming language specified by the instructor, convert the pseudocode for a selected process to program code, incorporating at least three of the following structures, the need for which will be dictated by the assigned problem(s) and process(es). The resulting code design can be event-driven, object-oriented, or procedural. a. Operations and functions (user-defined and/or library) b. Repetition (loops) c. Decision (if...else, case) d. Recursion	[15]		

Tennessee CS Standards Alignment with CodeX Curriculum			
Course Code: C10H14	Unit 1	Unit 2	Unit 3
10) Verify the correct operation of the resulting program code with several test cases: a. All valid values b. Error trapping of invalid values c. Error trapping of invalid program operation d. Troubleshooting/remedying program problems			
Project Planning and Quality Assurance			
11) Compile the necessary documentation to understand the nature of a computer programming problem and the customer/client specifications for the request and summarize in an informational text. This will include evidence of the scope of the problem, its attendant input and output information, the required system processing, and the software specifications involved.			
12) Analyze a given problem and develop a coherent strategy in the form of a project plan to meet the customer/client's need. The plan will include, but will not be limited to, defining the project scope as addressed by the problem documentation, identifying software development and implementation issues, timeline and benchmarks for design, and addressing issues associated with software maintenance and life cycle.			
13) In the software development process, articulate the nature of the program designs by creating documentation that addresses topics including but not limited to: a. The procedural, object-oriented, event-driven, or other nature of the various portions of the resulting application b. The data structures used for inputs, outputs, and internal manipulations c. The algorithms and guiding formulas used d. Constraints on accurate operation and results e. Modular designs that enable portability f. Interface details that permit ready maintenance and upkeep			
14) Apply principles of quality assurance during application development to certify bug tracking, audit trails, testing results, and other quality considerations. Annotate each quality assurance task with evidence from best practices endorsed by industry or research.			
15) Document the security risks associated with new applications and evaluate the severity of the risk involved in each, including but not limited to: a. Identifying threats to information systems facilities, data communications systems, and other applications b. Adhering to federal and state legislation pertaining to computer crime, fraud, and abuse c. Providing means for preserving confidentiality and encryption of sensitive data d. Detailing steps to recover from routine errors or catastrophic failures, such as might be caused by a malicious computer virus			

[1] These can be the remixes that begin in Mission 4

[2] This begins in Mission 4

[3] Mission 6 begins the use of code reuse

[4] 3.5 introduces the debugger

[5] Mission 2 and the teachers' manual discusses troubleshooting

[6] Flowcharts are introduced in the teachers' manual

[7] Mission 6 begins the use of nested loops but does not discuss them
4.7 begins the use of branching

[8] 3.8 introduces the use of variables

[9] Mission 4 introduces data types
Mission 3 introduces variable use
Mission 7 introduces the use of lists

[10] This can be done with the remixes depending on the rubric the teacher uses

[11] All of our missions require this to be able to progress to the next mission

[12] Pseudocodes and flowcharts are introduced in the teachers' manual

[13] 5.5 discusses descriptive variable naming

[14] These begin in Mission 3

[15] These begin with the remixes in Mission 4