CompuScholar, Inc.

Alignment to Tennessee **Programming & Logic I** Standards

Tennessee Course Details:

Course Name:	Programming & Logic I
Primary Career Cluster:	Information Technology
Course Code(s):	6098
Credit:	1
Grade Level:	10
Teacher Resources:	http://www.tn.gov/education/cte/InformationTechnology.shtml

CompuScholar Course Details:

Course Title: **TeenCoder: Windows Programming**
Course ISBN: **978-0-9887070-0-9**
Course Year: **2015**

Note 1: Standards were derived from this document, dated April 10, 2015:
http://www.tn.gov/education/cte/clusters/cte_std_programming_logic_1.pdf

Note 2: Citation(s) listed may represent a subset of the instances where objectives are met throughout the

Note 3: Additional requirements may be found in the Tennesee "Screening Instrument" designed for this course. Some specific "screening" requirements are cited at the end of this document. Other screening standards are met by the entire course as a whole, including all supplemental lessons (designed to meet unique state standards).

Course Description

Programming & Logic I is a course intended to teach students the basics of computer programming. The course places emphasis on practicing standard programming techniques and learning the logic tools and methods typically used by programmers to create simple computer applications. Upon completion of this course, proficient students will be able to solve problems by planning multistep procedures; write, analyze, review, and revise programs, converting detailed information from workflow charts and diagrams into coded instructions in a computer language; and will be able to troubleshoot/debug programs and software applications to correct malfunctions and ensure their proper execution. Standards in this course are aligned with the Tennessee State Standards for English Language Arts Standards and Literacy in Technical Subjects and Tennessee State Standards for Mathematics.*

Course Standards

Computer Programming Overview	CITATION(S)
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.	Chapter 1, Lesson 1
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.	Chapter 1, Lesson 4

Ethics	CITATION(S)
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.	Chapter 1, Lesson 5 (including research & discussion of EULA agreements)

Programming Skills	CITATION(S)
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.	Chapter 1, Lesson 3 differentiates between operating systems and applications, but n/a for the rest.
5) Apply the system management tools present in a programming development environment to:	See below
a. Select the most appropriate programming language for the task at hand	Chapter 1, Lesson 4
b. Develop syntactically correct program code using current best practices and emerging classes of development techniques	Students will write program code using best practices in nearly every chapter. See Chapter 2, Lesson 3 for discussion of comments & whitespace, Chapter 12 (Object-Oriented Programming), etc.

c. Use a compiler to interpret the source code and produce executable program code	Students will code, compile, and run programs in nearily every chapter. See Chapter 2, Lesson 1 for specific discussion of the compilation process.
6) In the process of developing and implementing programming solutions, develop strategies that work within the constraints of major operating system fundamentals, such as:	See below
a. Security protocols and procedures for accessing files and folders	Chapter 2, Lesson 2 (C# project files and folders) Students will also use OS-level tools (e.g. Windows Explorer) to copy/move starter files to a working directory for some projects (e.g. Chapter 7, Activity #1).
b. File management syntax requirements, including but not limited to creating, naming, organizing, copying, moving, and deleting files	Chapter 2, Lesson 2 describes the files and folders involved in a C# projects, how they are created, named, organized, etc. Students will use OS-level tools (e.g. web browsers, Windows Explorer) to download, un-ZIP, and copy/move starter files to a working directory for some projects (e.g. Chapter 7, Activity #1). Students will also identify, ZIP, and submit project folder & contents to the teacher in each activity.
c. File naming conventions, as they apply across multiple software applications and file types.	Chapter 2, Lesson 2 (File extensions) Supplemental Lesson 5 (File I/O)
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.	Chapter 7, Lesson 3 (Common Algorithms) Chapter 7, Activity 1 (Algorithms Practice)
8) Organize and develop a plan to acquire and manage the data values for a process, including the following:	See below
a. Data types, such as string, numeric, character, integer, and date	Chapter 4, Lesson 1
b. Program variable names	Chapter 4, Lesson 2

c. Variables and constants	Chapter 4, Lesson 2
d. Arrays (at least one- and two-dimensional), subscripts	Chapter 11, Lesson 1
e. Input from files and user responses	Supplemental Lesson 5 / Activity 5 (File I/O)
f. Output to files and reports	Supplemental Lesson 5 / Activity 5 (File I/O)
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.	Our hands-on programming projects include bulleted logic steps equivalent to pseudocode that students translate into C# programs. See below for details.
a. Operations and functions (user-defined and/or library)	Chapter 7, Lesson 2 (.NET Math functions) Chapter 9 (student-written methods) Chapter 9 Activity (What's Your Birthday)
b. Repetition (loops)	Chapter 5, Lesson 3 (For loops) Chapter 5, Lesson 4 (While loops) Chapter 5 Activity (Jeepers Beepers)
c. Decision (if…else, case)	Chapter 5, Lesson 2 (if…else) Chapter 5 Activity (Jeepers Beeprs)
d. Recursion	Chapter 14, Lesson 2 (Recursion) Chapter 14, Activity 2 (Finding Fibonacco Numbers)
10) Verify the correct operation of the resulting program code with several test cases:	See below
a. All valid values	Chapter 10, Lesson 4
b. Error trapping of invalid values	Chapter 10, Lesson 4 Chapter 5 Activity (Jeepers Beepers)
c. Error trapping of invalid program operation	Chapter 10, Lesson 3

d. Troubleshooting/remedying program problems	Chapter 10, Lessons 1, 2, 4 Chapter 10 Activity (Divide By Zero)

Project Planning and Quality Assurance	CITATION(S)
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.	Supplemental Lesson 6 / Activity 6
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.	Supplemental Lesson 6 / Activity 6
13) In the software development process, articulate the nature of the program designs by creating documentation that addresses topics including but not limited to:	See below
a. The procedural, object-oriented, event-driven, or other nature of the various portions of the resulting application	Supplemental Lesson 6 / Activity 6
b. The data structures used for inputs, outputs, and internal manipulations	Supplemental Lesson 6 / Activity 6
c. The algorithms and guiding formulas used	Supplemental Lesson 6 / Activity 6
d. Constraints on accurate operation and results	Supplemental Lesson 6 / Activity 6
e. Modular designs that enable portability	Supplemental Lesson 6 / Activity 6
f. Interface details that permit ready maintenance and upkeep	Supplemental Lesson 6 / Activity 6
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.	Supplemental Lesson 6 / Activity 6
15) Document the security risks associated with new applications and evaluate the severity of the risk involved in each, including but not limited to:	n/a
a. Identifying threats to information systems facilities, data communications systems, and other applications	n/a
b. Adhering to federal and state legislation pertaining to computer crime, fraud, and abuse	n/a

c. Providing means for preserving confidentiality and encryption of sensitive data	n/a
d. Detailing steps to recover from routine errors or catastrophic failures, such as might be caused by a malicious computer virus	n/a

From SECTION I(3) of the "Programming & Software Development Screening Instrument1-9-15": POSTSECONDARY AND CAREER READINESS	CITATION(S)
A. Technical skills are promoted within the context of applicable industries and work environments. They are not presented in isolation or without meaningful connections to aligned careers.	Supplemental Lesson 3 / Activity 3 (Software Development Careers / Career Exporation Exercise) Supplemental Lesson 6 / Activity 6 (Software Development Process / Your SDLC Docs)
B. Materials showcase a diversity of career and postsecondary opportunities for students upon completion of high school, including all applicable levels of postsecondary training (i.e., technical schools, community colleges, four-year universities, etc.).	Supplemental Lesson 3 / Activity 3 (Software Development Careers / Career Exporation Exercise)
C. Connections to relevant certifications and other credentials are clearly explained, and their value in industry is communicated where appropriate.	Supplemental Lesson 3 / Activity 3 (Software Development Careers / Career Exporation Exercise)
D. Materials provide opportunities for students to practice and reflect upon 21st century (or “soft”) skills.	Supplemental Lesson 2 / Activity 2 (Stock Market Trading) Supplemental Lesson 3 / Activity 3 (Software Development Careers / Career Exporation Exercise) Supplemental Lesson 6 / Activity 6 (Software Development Process / Your SDLC Docs)