Computer Science I
At-A-Glance - Lamar CISD

Ongoing Skills Imbedded All Year	**Professional Standards/Employability Skills/Technical Skills**		
	● Increase and diversify participation in computer science ● Students, regardless of prior experience in computing, will develop confidence using computer science as a tool to express themselves and solve problems, and this confidence will prepare them for success in future endeavors in the field of computer science ● Students will understand the core principles of computing, a field which has and continues to change the world ● Students will be able to develop computational artifacts to solve problems, communicate ideas, and express their own creativity ● Students will be able to collaborate with others to solve problems and develop computational artifacts 1● Students will be able to explain the impact computing has on society, economy, and culture ● Students will be able to analyze existing artifacts, identify and correct errors, and explain how the artifact functions ● Students will be able to explain how data, information, or knowledge is represented for computational use ● Students will be able to explain how abstractions are used in computation and modeling ● Students will learn to be informed and responsible users of technology		
Ongoing Ways to Show	Complete labs and assignment individually or as a team.		
Grading Period	**Unit Name**	**Estimated Time Frame**	**TEKS**
Grading Period 1 **29 Days**	**Introductory Skills/Set Up**	**2 Days**	2H
	2(H) The student will seek and respond to advice from peers and professionals in evaluating quality and accuracy.		
	Unit 1: Welcome	**2 Days**	6A, 6B, 6C, 6D, 6V
	6(A) The student will compare and contrast types of operating systems, software applications, and programming languages. 6(B) The student will demonstrate knowledge of major hardware components, including primary and secondary memory, a central processing unit (CPU), and peripherals. 6(C) The student will differentiate among current programming languages, discuss the use of those languages in other fields of study, and demonstrate knowledge of specific programming terminology and concepts. 6(D) The student will differentiate between a high-level compiled language and an interpreted language. 6(V) The student will compare and contrast strongly typed and un-typed programming languages.		
	Unit 2: Basic Python and Console Interaction	**15 Days**	2A, 2B, 2D, 2R, 4D, 4F, 4H, 4I, 4J ,4O, 4Q, 4R, 6O, 6P, 6R
	2(A) Create and properly display meaningful output. 2(B) Create interactive console display interfaces, with appropriate user prompts, to acquire data from a user. 2(D) Write programs and communicate with proper programming style to enhance the readability and functionality of the code by using meaningful descriptive identifiers, internal comments, white space, indentation, and a standardized program style. 2(R) Develop sequential algorithms to solve non-branching and non-iterative problems. 4(D) Identify the data types and objects needed to solve a problem. 4(F) Design a solution to a problem. 4(H) Identify and debug errors. 4(I) Test program solutions with appropriate valid and invalid test data for correctness. 4(J) Debug and solve problems using error messages, reference materials, language documentation, and effective strategies. 4(O) Demonstrate proficiency in the use of the arithmetic operators to create mathematical expressions, including addition, subtraction, multiplication, real division, integer division, and modulus division. 4(Q) Develop program solutions that use assignment. 4(R) Develop sequential algorithms to solve non-branching and non-iterative problems. 6(O) Choose, identify, and use the appropriate data types for integer, real, and Boolean data when writing program solutions. 6(P) Demonstrate an understanding of the concept of a variable. 6(R) Demonstrate an understanding of how to represent and manipulate text data, including concatenation and other string functions.		

	Unit 3: Introduction to Programming with Turtle Graphics	**10 Days**	2A, 2B, 2D, 2F, 4A, 4B, 4C, 4D, 4E, 4F, 4G, 4I, 4N, 4O, 4Q, 4R, 4S, 4T, 6H, 6P
	2(A) Create and properly display meaningful output. 2(B) Create interactive console display interfaces, with appropriate user prompts, to acquire data from a user. 2(D) Write programs with proper programming style to enhance the readability and functionality of the code by using meaningful descriptive identifiers, internal comments, white space, spacing, indentation, and a standardized program style. 2(F) Display simple vector graphics using lines, circles, and rectangles. 4(A) Use program design problem-solving strategies to create program solutions. 4(B) Define and specify the purpose and goals of solving a problem. 4(C) Identify the subtasks needed to solve a problem. 4(D) Identify the data types and objects needed to solve a problem. 4(E) Identify reusable components from existing code. 4(F) Design a solution to a problem. 4(G) Code a solution from a program design. 4(I) Analyze and modify existing code to improve the underlying algorithm. 4(N) Select the most appropriate algorithm for a defined problem. 4(O) Demonstrate proficiency in the use of the arithmetic operators to create mathematical expressions, including addition, subtraction, multiplication, real division, integer division, and modulus division. 4(Q) Develop program solutions that use assignment. 4(R) Develop sequential algorithms to solve non-branching and non-iterative problems. 4(S) Develop algorithms to decision-making problems using branching control statements. 4(T) Develop iterative algorithms and code programs to solve practical problems. 6(H) Create subroutines that do not return values with and without the use of arguments and parameters. 6(P) Demonstrate an understanding of the concept of a variable.		
Grading Period 2 **27 Days**	**Unit 3: Introduction to Programming with Turtle Graphics**	**14 Days**	2A, 2B, 2D, 2F, 4A, 4B, 4C, 4D, 4E, 4F, 4G, 4I, 4N, 4O, 4Q, 4R, 4S, 4T, 6H, 6P
	2(A) Create and properly display meaningful output. 2(B) Create interactive console display interfaces, with appropriate user prompts, to acquire data from a user. 2(D) Write programs with proper programming style to enhance the readability and functionality of the code by using meaningful descriptive identifiers, internal comments, white space, spacing, indentation, and a standardized program style. 2(F) Display simple vector graphics using lines, circles, and rectangles. 4(A) Use program design problem-solving strategies to create program solutions. 4(B) Define and specify the purpose and goals of solving a problem. 4(C) Identify the subtasks needed to solve a problem. 4(D) Identify the data types and objects needed to solve a problem. 4(E) Identify reusable components from existing code. 4(F) Design a solution to a problem. 4(G) Code a solution from a program design. 4(I) Analyze and modify existing code to improve the underlying algorithm. 4(N) Select the most appropriate algorithm for a defined problem. 4(O) Demonstrate proficiency in the use of the arithmetic operators to create mathematical expressions, including addition, subtraction, multiplication, real division, integer division, and modulus division. 4(Q) Develop program solutions that use assignment. 4(R) Develop sequential algorithms to solve non-branching and non-iterative problems. 4(S) Develop algorithms to decision-making problems using branching control statements. 4(T) Develop iterative algorithms and code programs to solve practical problems. 6(H) Create subroutines that do not return values with and without the use of arguments and parameters. 6(P) Demonstrate an understanding of the concept of a variable.		
	Unit 4: Conditionals	**13 Days**	2A, 2B, 4F, 4H, 4I, 4J, 4O, 4P, 4Q, 4S, 4U, 4V
	2(A) Create and properly display meaningful output. 2(B) Create interactive console display interfaces, with appropriate user prompts, to acquire data from a user. 4(F) Design a solution to a problem. 4(H) Identify and debug errors. 4(I) Test program solutions with appropriate valid and invalid test data for correctness. 4(J) Debug and solve problems using error messages, reference materials, language documentation, and effective strategies. 4(O) Demonstrate proficiency in the use of the arithmetic operators to create mathematical expressions, including addition, subtraction, multiplication, real division, integer division, and modulus division. 4(P) Create program solutions to problems using available mathematics libraries, including absolute value, round, power, square, and square root.		

Computer Science I
Lab Safety and Scientific Processes
Readiness Standards
Supporting Standards

	4(Q) Develop program solutions that use assignment. 4(S) Develop algorithms to decision-making problems using branching control statements. 4(U) Demonstrate proficiency in the use of the relational operators. 4(V) Demonstrate proficiency in the use of the logical operators.		
Grading Period 3 **28 Days**	**Unit 4: Conditionals**	**16 Days**	2A, 2B, 4F, 4H, 4I, 4J, 4O, 4P, 4Q, 4S, 4U, 4V
	2(A) Create and properly display meaningful output. 2(B) Create interactive console display interfaces, with appropriate user prompts, to acquire data from a user. 4(F) Design a solution to a problem. 4(H) Identify and debug errors. 4(I) Test program solutions with appropriate valid and invalid test data for correctness. 4(J) Debug and solve problems using error messages, reference materials, language documentation, and effective strategies. 4(O) Demonstrate proficiency in the use of the arithmetic operators to create mathematical expressions, including addition, subtraction, multiplication, real division, integer division, and modulus division. 4(P) Create program solutions to problems using available mathematics libraries, including absolute value, round, power, square, and square root. 4(Q) Develop program solutions that use assignment. 4(S) Develop algorithms to decision-making problems using branching control statements. 4(U) Demonstrate proficiency in the use of the relational operators. 4(V) Demonstrate proficiency in the use of the logical operators.		
	Unit 5: Looping	**12 Days**	2A, 2B, 4F, 4I, 4K, 4L, 4N, 4O, 4P, 4Q, 4S, 4T
	2(A) Create and properly display meaningful output. 2(B) Create interactive console display interfaces, with appropriate user prompts, to acquire data from a user. 4(F) Design a solution to a problem 4(I) Test program solutions with appropriate valid and invalid test data for correctness. 4(K) Explore common algorithms, including finding greatest common divisor, finding the biggest number out of three, finding primes, making change, and finding the average. 4(L) Analyze and modify existing code to improve the underlying algorithm. 4(N) Select the most appropriate algorithm for a defined problem. 4(O) Demonstrate proficiency in the use of the arithmetic operators to create mathematical expressions, including addition, subtraction, multiplication, real division, integer division, and modulus division. 4(P) Create program solutions to problems using available mathematics libraries, including absolute value, round, power, square, and square root. 4(Q) Develop program solutions that use assignment. 4(S) Develop algorithms to decision-making problems using branching control statements. 4(T) Develop iterative algorithms and code programs to solve practical problems.		
Grading Period 4 **31 Days**	**Unit 6: Functions and Exceptions**	**16 Days**	2A, 2B, 4E, 4F, 4H, 4I, 4J, 4L, 4M, 4O, 4Q, 4S, 4T, 6F, 6H, 6I, 6S
	2(A) Create and properly display meaningful output. 2(B) Create interactive console display interfaces, with appropriate user prompts, to acquire data from a user. 4(E) Identify reusable components from existing code. 4(F) Design a solution to a problem 4(H) Identify and debug errors. 4(I) Test program solutions with appropriate valid and invalid test data for correctness. 4(J) Debug and solve problems using error messages, reference materials, language documentation, and effective strategies. 4(L) Analyze and modify existing code to improve the underlying algorithm. 4(M) Create program solutions that exhibit robust behavior by understanding, avoiding, and preventing runtime errors, including division by zero and type mismatch. 4(O) Demonstrate proficiency in the use of the arithmetic operators to create mathematical expressions, including addition, subtraction, multiplication, real division, integer division, and modulus division. 4(Q) Develop program solutions that use assignment. 4(S) Develop algorithms to decision-making problems using branching control statements. 4(T) Develop iterative algorithms and code programs to solve practical problems. 6(F) Use local and global scope access variable declarations. 6(H) Create subroutines that do not return values with and without the use of arguments and parameters. 6(I) Create subroutines that return typed values with and without the use of arguments and parameters. 6(S) Demonstrate an understanding of the concept of scope.		

	Unit 7: Strings	**15 Days**	2A, 2B, 4F, 4H, 4I, 4J, 4O, 4Q, 4S, 4T, 6R
	2(A) Create and properly display meaningful output. 2(B) Create interactive console display interfaces, with appropriate user prompts, to acquire data from a user. 4(F) Design a solution to a problem. 4(H) Identify and debug errors. 4(I) Test program solutions with appropriate valid and invalid test data for correctness. 4(J) Debug and solve problems using error messages, reference materials, language documentation, and effective strategies. 4(O) Demonstrate proficiency in the use of the arithmetic operators to create mathematical expressions, including addition, subtraction, multiplication, real division, integer division, and modulus division. 4(Q) Develop program solutions that use assignment. 4(S) Develop algorithms to decision-making problems using branching control statements. 4(T) Develop iterative algorithms and code programs to solve practical problems. 6(R) Demonstrate an understanding of how to represent and manipulate text data, including concatenation and other string functions.		
Grading Period 5 **30 Days**	**Unit 8: Creating and Altering Data Structures**	**15 Days**	2A, 2B, 4D, 4F, 4I, 4K, 4O, 4P, 4Q, 4S, 4T, 6T, 6U
	2(A) Create and properly display meaningful output. 2(B) Create interactive console display interfaces, with appropriate user prompts, to acquire data from a user. 4(D) Identify the data types and objects needed to solve a problem. 4(F) Design a solution to a problem. 4(I) Test program solutions with appropriate valid and invalid test data for correctness. 4(K) Explore common algorithms, including finding greatest common divisor, finding the biggest number out of three, finding primes, making change, and finding the average. 4(O) Demonstrate proficiency in the use of the arithmetic operators to create mathematical expressions, including addition, subtraction, multiplication, real division, integer division, and modulus division. 4(P) Create program solutions to problems using available mathematics libraries, including absolute value, round, power, square, and square root. 4(Q) Develop program solutions that use assignment. 4(S) Develop algorithms to decision-making problems using branching control statements. 4(T) Develop iterative algorithms and code programs to solve practical problems, 6(T) Identify and use the structured data type of one-dimensional arrays to traverse, search, and modify data. 6(U) Choose, identify, and use the appropriate data type and structure to properly represent the data in a program problem solution.		
	Unit 9: Extending Data Structure.	**15 Days**	2A, 4D, 4F, 4G, 4O, 4Q, 4R, 4S, 4T, 6U
	2(A) Create and properly display meaningful output. 4(D) Identify the data types and objects needed to solve a problem. 4(F) Design a solution to a problem. 4(G) Code a solution from a program design. 4(O) Demonstrate proficiency in the use of the arithmetic operators to create mathematical expressions, including addition, subtraction, multiplication, real division, integer division, and modulus division. 4(Q) Develop program solutions that use assignment. 4(R) Develop sequential algorithms to solve non-branching and non-iterative problems. 4(S) Develop algorithms to decision-making problems using branching control statements. 4(T) Develop iterative algorithms and code programs to solve practical problems. 6(U) Choose, identify, and use the appropriate data type and structure to properly represent the data in a program problem solution.		
Grading Period 6 **27 Days**	**CodeHS Certification**	**10 Days**	
	Study Units and Certification Test		
	End-of-year project	**17 Days**	4A, 4K, 6T
	4(A) The student will use program design problem-solving strategies to create program solutions. 4(K) The student will explore common algorithms, including finding greatest common divisor, finding the biggest number out of three, finding primes, making change, and finding the average. 6(T) The student will identify and use the structured data type of one-dimensional arrays to traverse, search, and modify data.		

Computer Science I
Lab Safety and Scientific Processes
Readiness Standards
Supporting Standards