§126.34. Computer Science II TEKS

(a) General requirements. Students shall be awarded one credit for successful completion of this course. The required prerequisites for this course are Algebra I and either Computer Science I or Fundamentals of Computer Science. This course is recommended for students in Grades 11 and 12.

(b) Introduction.

(1) The technology applications curriculum has six strands based on the National Educational Technology Standards for Students (NETS•S) and performance indicators developed by the International Society for Technology in Education (ISTE): creativity and innovation; communication and collaboration; research and information fluency; critical thinking, problem solving, and decision making; digital citizenship; and technology operations and concepts.

(2) Computer Science II will foster students' creativity and innovation by presenting opportunities to design, implement, and present meaningful programs through a variety of media. Students will collaborate with one another, their instructor, and various electronic communities to solve the problems presented throughout the course. Through data analysis, students will identify task requirements, plan search strategies, and use computer science concepts to access, analyze, and evaluate information needed to solve problems. By using computer science knowledge and skills that support the work of individuals and groups in solving problems, students will select the technology appropriate for the task, synthesize knowledge, create solutions, and evaluate the results. Students will learn digital citizenship by researching current laws and regulations and by practicing integrity and respect. Students will gain an understanding of computer science through the study of technology operations, systems, and concepts.

(3) Statements that contain the word "including" reference content that must be mastered, while those containing the phrase "such as" are intended as possible illustrative examples.

(c) Knowledge and skills.

(1) Creativity and innovation. The student develops products and generates new understandings by extending existing knowledge. The student is expected to:

(A) use program design problem-solving strategies to create program solutions;

(B) demonstrate the ability to read and modify large programs, including the design description and process development;

(C) follow the systematic problem-solving process of identifying the specifications of purpose and goals, the data types and objects needed, and the subtasks to be performed;

(D) compare and contrast design methodologies and implementation techniques such as top-down, bottom-up, and black box;

(E) analyze, modify, and evaluate existing code by performing a case study on a large program, including inheritance and black box programming;

(F) identify the data types and objects needed to solve a problem;

(G) choose, identify, and use the appropriate abstract data type, advanced data structure, and supporting algorithms to properly represent the data in a program problem solution;

(H) use object-oriented programming development methodology, data abstraction, encapsulation with information hiding, and procedural abstraction in program development and testing; and

(I) create, edit, and manipulate bitmap images that are used to enhance user interfaces and program functionality.

(2) Communication and collaboration. The student communicates and collaborates with peers to contribute to his or her own learning and the learning of others. The student is expected to:

(A) use the principles of software engineering to work in software design teams, break a problem statement into specific solution requirements, create a program development plan, code part of a solution from a program development plan while a partner codes the remaining part, team test the solution for correctness, and develop presentations to report the solution findings;

(B) create interactive console display interfaces with appropriate user prompts;

(C) create interactive human interfaces to acquire data from a user and display program results using an advanced Graphical User Interface (GUI);

(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;

(E) improve data display by optimizing data visualization;

(F) display simple vector graphics to interpret and display program results; and

(G) display simple bitmap images.

(3) Research and information fluency. The student locates, analyzes, processes, and organizes data. The student is expected to:

(A) use local area networks (LANs) and wide area networks (WANs), including the Internet and intranets, in research, file management, and collaboration;

(B) understand programming file structure and file access for required resources;

(C) acquire and process information from text files, including files of known and unknown sizes;

(D) manipulate data structures using string processing;

(E) manipulate data values by casting between data types;

(F) identify and use the structured data type of one-dimensional arrays to traverse, search, modify, insert, and delete data;

(G) identify and use the structured data type of two-dimensional arrays to traverse, search, modify, insert, and delete data; and

(H) identify and use a list object data structure to traverse, search, insert, and delete data.

(4) Critical thinking, problem solving, and decision making. The student uses appropriate strategies to analyze problems and design algorithms. The student is expected to:

(A) develop sequential algorithms using branching control statements, including nested structures, to create solutions to decision-making problems;

(B) develop choice algorithms using selection control statements based on ordinal values;

(C) demonstrate proficiency in the use of short-circuit evaluation;

(D) demonstrate proficiency in the use of Boolean algebra, including De Morgan's Law;

(E) develop iterative algorithms using nested loops;

(F) identify, trace, and appropriately use recursion in programming solutions, including algebraic computations;

(G) design, construct, evaluate, and compare search algorithms, including linear searching and binary searching;

(H) identify, describe, design, create, evaluate, and compare standard sorting algorithms, including selection sort, bubble sort, insertion sort, and merge sort;

(I) measure time/space efficiency of various sorting algorithms;

(J) compare and contrast search and sort algorithms, including linear, quadratic, and recursive strategies, for time/space efficiency;

(K) analyze algorithms using "big-O" notation for best, average, and worst-case data patterns;

(L) develop algorithms to solve various problems, including factoring, summing a series, finding the roots of a quadratic equation, and generating Fibonacci numbers;

(M) test program solutions by investigating boundary conditions; testing classes, methods, and libraries in isolation; and performing stepwise refinement;

(N) identify and debug compile, syntax, runtime, and logic errors;

(O) compare and contrast algorithm efficiency by using informal runtime comparisons, exact calculation of statement execution counts, and theoretical efficiency values using "big-O" notation, including worst-case, best-case, and average-case time/space analysis;

(P) demonstrate the ability to count, convert, and perform mathematical operations in the binary and hexadecimal number systems;

(Q) demonstrate knowledge of the maximum integer boundary, minimum integer boundary, imprecision of real number representations, and round-off errors;

(R) create program solutions to problems using the mathematics library class;

(S) use random algorithms to create simulations that model the real world;

(T) identify, understand, and create class specifications and relationships among classes, including composition and inheritance relationships;

(U) understand and explain object relationships among defined classes, abstract classes, and interfaces;

(V) create object-oriented definitions using class declarations, variable declarations, constant declarations, method declarations, parameter declarations, and interface declarations;

(W) create robust classes that encapsulate data and the methods that operate on that data and incorporate overloading to enrich the object's behavior;

(X) design and implement a set of interactive classes;

(Y) design, create, and evaluate multiclass programs that use abstract classes and interfaces;

(Z) understand and implement a student-created class hierarchy;

(AA) extend, modify, and improve existing code using inheritance;

(BB) create adaptive behaviors, including overloading, using polymorphism;

(CC) understand and use reference variables for object and string data types;

(DD) understand and implement access scope modifiers;

(EE) understand and demonstrate how to compare objects;

(FF) duplicate objects using the appropriate deep and/or shallow copy;

(GG) define and implement abstract classes and interfaces in program problem solutions;

(HH) apply functional decomposition to a program solution;

(II) create simple and robust objects from class definitions through instantiation;

(JJ) apply class membership of variables, constants, and methods;

(KK) examine and mutate the properties of an object using accessors and modifiers;

(LL) understand and implement a composite class; and

(MM) design and implement an interface.

(5) Digital citizenship. The student explores and understands safety, legal, cultural, and societal issues relating to the use of technology and information. The student is expected to:

(A) model ethical acquisition and use of digital information;

(B) demonstrate proper digital etiquette, responsible use of software, and knowledge of acceptable use policies; and

(C) investigate digital rights management.

(6) Technology operations and concepts. The student understands technology concepts, systems, and operations as they apply to computer science. The student is expected to:

(A) compare and contrast types of operating systems, software applications, hardware platforms, and programming languages;

(B) demonstrate knowledge of major hardware components, including primary and secondary memory, a central processing unit (CPU), and peripherals;

(C) demonstrate knowledge of major networking components, including hosts, servers, switches, and routers;

(D) demonstrate knowledge of computer communication systems, including single-user, peer-to-peer, workgroup, client-server, and networked;

(E) demonstrate knowledge of computer addressing systems, including Internet Protocol (IP) address and Media Access Control (MAC) address; and

(F) differentiate among the categories of programming languages, including machine, assembly, high-level compiled, high-level interpreted, and scripted.

Source: The provisions of this §126.34 adopted to be effective September 26, 2011, 36 TexReg 6263.