PEIMS #: 03580300

Implementation Year: 2012-13

| **Knowledge & Skills** | **Student Expectation** | **Bloom's Level** | **TEKS Alignment Evidence** | **Recommendations** |
| --- | --- | --- | --- | --- |
| **Guideline** | **Observed**  |
| (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; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (B) demonstrate the ability to read and modify large programs, including the design description and process development; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (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; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (D) compare and contrast design methodologies and implementation techniques such as top-down, bottom-up, and black box; | Analyze | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (E) analyze, modify, and evaluate existing code by performing a case study on a large program, including inheritance and black box programming; | Evaluate | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (F) identify the data types and objects needed to solve a problem; | Understand | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (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; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (H) use object-oriented programming development methodology, data abstraction, encapsulation with information hiding, and procedural abstraction in program development and testing; and | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (I) create, edit, and manipulate bitmap images that are used to enhance user interfaces and program functionality. | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (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; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (B) create interactive console display interfaces with appropriate user prompts; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (C) create interactive human interfaces to acquire data from a user and display program results using an advanced Graphical User Interface (GUI); | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (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; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (E) improve data display by optimizing data visualization; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (F) display simple vector graphics to interpret and display program results; and | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (G) display simple bitmap images. | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (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; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (B) understand programming file structure and file access for required resources; | Understand | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (C) acquire and process information from text files, including files of known and unknown sizes; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (D) manipulate data structures using string processing; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (E) manipulate data values by casting between data types; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (F) identify and use the structured data type of one-dimensional arrays to traverse, search, modify, insert, and delete data; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (G) identify and use the structured data type of two-dimensional arrays to traverse, search, modify, insert, and delete data; and | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (H) identify and use a list object data structure to traverse, search, insert, and delete data. | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (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; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (B) develop choice algorithms using selection control statements based on ordinal values; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (C) demonstrate proficiency in the use of short-circuit evaluation; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (D) demonstrate proficiency in the use of Boolean algebra, including De Morgan's Law; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (E) develop iterative algorithms using nested loops; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (F) identify, trace, and appropriately use recursion in programming solutions, including algebraic computations; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (G) design, construct, evaluate, and compare search algorithms, including linear searching and binary searching; | Analyze | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (H) identify, describe, design, create, evaluate, and compare standard sorting algorithms, including selection sort, bubble sort, insertion sort, and merge sort; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (I) measure time/space efficiency of various sorting algorithms; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (J) compare and contrast search and sort algorithms, including linear, quadratic, and recursive strategies, for time/space efficiency; | Analyze | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (K) analyze algorithms using "big-O" notation for best, average, and worst-case data patterns; | Analyze | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (L) develop algorithms to solve various problems, including factoring, summing a series, finding the roots of a quadratic equation, and generating Fibonacci numbers; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (M) test program solutions by investigating boundary conditions; testing classes, methods, and libraries in isolation; and performing stepwise refinement; | Analyze | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (N) identify and debug compile, syntax, runtime, and logic errors; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (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; | Analyze | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (P) demonstrate the ability to count, convert, and perform mathematical operations in the binary and hexadecimal number systems; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (Q) demonstrate knowledge of the maximum integer boundary, minimum integer boundary, imprecision of real number representations, and round-off errors; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (R) create program solutions to problems using the mathematics library class; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (S) use random algorithms to create simulations that model the real world; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (T) identify, understand, and create class specifications and relationships among classes, including composition and inheritance relationships; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (U) understand and explain object relationships among defined classes, abstract classes, and interfaces; | Understand | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (V) create object-oriented definitions using class declarations, variable declarations, constant declarations, method declarations, parameter declarations, and interface declarations; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (W) create robust classes that encapsulate data and the methods that operate on that data and incorporate overloading to enrich the object's behavior; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (X) design and implement a set of interactive classes; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (Y) design, create, and evaluate multiclass programs that use abstract classes and interfaces; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (Z) understand and implement a student-created class hierarchy; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (AA) extend, modify, and improve existing code using inheritance; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (BB) create adaptive behaviors, including overloading, using polymorphism; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (CC) understand and use reference variables for object and string data types; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (DD) understand and implement access scope modifiers; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (EE) understand and demonstrate how to compare objects; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (FF) duplicate objects using the appropriate deep and/or shallow copy; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (GG) define and implement abstract classes and interfaces in program problem solutions; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (HH) apply functional decomposition to a program solution; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (II) create simple and robust objects from class definitions through instantiation; | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (JJ) apply class membership of variables, constants, and methods; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (KK) examine and mutate the properties of an object using accessors and modifiers; | Analyze | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (LL) understand and implement a composite class; and | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (MM) design and implement an interface. | Create | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (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; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (B) demonstrate proper digital etiquette, responsible use of software, and knowledge of acceptable use policies; and | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (C) investigate digital rights management. | Analyze | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (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; | Analyze | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (B) demonstrate knowledge of major hardware components, including primary and secondary memory, a central processing unit (CPU), and peripherals; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (C) demonstrate knowledge of major networking components, including hosts, servers, switches, and routers; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (D) demonstrate knowledge of computer communication systems, including single-user, peer-to-peer, workgroup, client-server, and networked; | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (E) demonstrate knowledge of computer addressing systems, including Internet Protocol (IP) address and Media Access Control (MAC) address; and | Apply | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |
| (F) differentiate among the categories of programming languages, including machine, assembly, high-level compiled, high-level interpreted, and scripted. | Analyze | **Select One:**RememberUnderstandApplyAnalyzeEvaluateCreate | **Select One:**ObservedPartially ObservedNot Observed **Where**: |  |

|  |
| --- |
| **Course Strengths of TEKS Alignment** |
|  |

|  |
| --- |
| **Required Changes for TEKS Alignment**(List the missing and partially covered TEKS ~AND~ summarize required TEKS modification, big or small, you believe must be completed prior to course approval) |
| Missing or Partially covered TEKS: TEKS Summary:  |

|  |
| --- |
| **Additional Opportunities for Improvement-Optional** (Non-required additions or changes that would improve this course) |
|  |