Computer Science I

Subject: Career Development and Career and Technical Education
Grade: 09
Expectations: 62
Breakouts: 186

(a) Introduction.

1. Career and technical education instruction provides content aligned with challenging academic standards, industry-relevant technical knowledge, and college and career readiness skills for students to further their education and succeed in current and emerging professions
2. The Science, Technology, Engineering, and Mathematics (STEM) Career Cluster focuses on planning, managing, and providing scientific research and professional and technical services such as laboratory and testing services and research and development services
3. Computer Science I 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 computational thinking and 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, regulations, and best practices and by practicing integrity and respect. Students will gain an understanding of the principles of computer science through the study of technology operations, systems, and concepts
4. Students are encouraged to participate in extended learning experiences such as career and technical student organizations and other leadership or extracurricular organizations
5. 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

(b) Knowledge and Skills Statements

(1) Employability. The student identifies various employment opportunities in the computer science field. The student is expected to:

(A) identify job and internship opportunities and accompanying job duties and tasks and contact one or more companies or organizations to explore career opportunities;

(i) identify job opportunities

(ii) identify internship opportunities

(iii) identify accompanying job duties

(iv) identify accompanying job tasks

(v) contact one or more companies or organizations to explore career opportunities

(B) examine the role of certifications, resumes, and portfolios in the computer science profession;

(i) examine the role of certifications in the computer science profession

(ii) examine the role of resumes in the computer science profession

(iii) examine the role of portfolios in the computer science profession

(C) employ effective technical reading and writing skills;

(i) employ effective technical reading skills

(ii) employ effective technical writing skills

(D) employ effective verbal and non-verbal communication skills;

(i) employ effective verbal communication skills

(ii) employ effective non-verbal communication skills

(E) solve problems and think critically;

(i) solve problems

(ii) think critically

(F) demonstrate leadership skills and function effectively as a team member;

(i) demonstrate leadership skills

(ii) function effectively as a team member

(G) communicate an understanding of legal and ethical responsibilities in relation to the field of computer science;

(i) communicate an understanding of legal responsibilities in relation to the field of computer science

(ii) communicate an understanding of ethical responsibilities in relation to the field of computer science

(H) demonstrate planning and time-management skills; and

(i) demonstrate planning skills

(ii) demonstrate time-management skills

(I) compare university computer science programs.

(i) compare university computer science programs

(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) participate in learning communities as a learner, initiator, contributor, and teacher/mentor; and

(i) participate in learning communities as a learner

(ii) participate in learning communities as a initiator

(iii) participate in learning communities as a contributor

(iv) participate in learning communities as a teacher

(v) participate in learning communities as a mentor

(B) seek and respond to advice from peers, educators, or professionals when evaluating quality and accuracy of the student's product.

(i) seek to advice from peers, educators, or professionals when evaluating quality of the student's product

(ii) seek to advice from peers, educators, or professionals when evaluating accuracy of the student's product

(iii) respond to advice from peers, educators, or professionals when evaluating quality of the student's product

(iv) respond to advice from peers, educators, or professionals when evaluating accuracy of the student's product

(3) Programming style and presentation. The student utilizes proper programming style and develops appropriate visual presentation of data, input, and output. The student is expected to:

(A) create and properly label and display output;

(i) create output

(ii) properly label output

(iii) properly display output

(B) create interactive input interfaces, with relevant user prompts, to acquire data from a user such as console displays or Graphical User Interfaces (GUIs);

(i) create interactive input interfaces, with relevant user prompts, to acquire data from a user

(C) write programs with proper programming style to enhance the readability and functionality of a code by using descriptive identifiers, internal comments, white space, spacing, indentation, and a standardized program style;

(i) write programs with proper programming style to enhance the readability of a code by using descriptive identifiers

(ii) write programs with proper programming style to enhance the readability of a code by using internal comments

(iii) write programs with proper programming style to enhance the readability of a code by using white space

(iv) write programs with proper programming style to enhance the readability of a code by using spacing

(v) write programs with proper programming style to enhance the readability of a code by using indentation

(vi) write programs with proper programming style to enhance the readability of a code by using a standardized program style

(vii) write programs with proper programming style to enhance the functionality of a code by using descriptive identifiers

(viii) write programs with proper programming style to enhance the functionality of a code by using internal comments

(ix) write programs with proper programming style to enhance the functionality of a code by using white space

(x) write programs with proper programming style to enhance the functionality of a code by using spacing

(xi) write programs with proper programming style to enhance the functionality of a code by using indentation

(xii) write programs with proper programming style to enhance the functionality of a code by using a standardized program style

(D) format data displays using standard formatting styles; and

(i) format data displays using standard formatting styles

(E) display simple vector graphics using lines, circles, and rectangles.

(i) display simple vector graphics using lines

(ii) display simple vector graphics using circles

(iii) display simple vector graphics using rectangles

(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) use program design problem-solving strategies such as flowchart or pseudocode to create program solutions;

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

(B) create a high-level program plan using a visual tool such as a flowchart or graphic organizer;

(i) create a high-level program plan using a visual tool

(C) identify the tasks and subtasks needed to solve a problem;

(i) identify the tasks needed to solve a problem

(ii) identify the subtasks needed to solve a problem

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

(i) identify the data types needed to solve a problem

(ii) identify the objects needed to solve a problem

(E) identify reusable components from existing code;

(i) identify reusable components from existing code

(F) design a solution to a problem;

(i) design a solution to a problem

(G) code a solution from a program design;

(i) code a solution from a program design

(H) identify error types, including syntax, lexical, run time, and logic;

(i) identify error types, including syntax

(ii) identify error types, including lexical

(iii) identify error types, including run time

(iv) identify error types, including logic

(I) test program solutions with valid and invalid test data and analyze resulting behavior;

(i) test program solutions with valid test data

(ii) test program solutions with invalid test data

(iii) analyze resulting behavior [after using valid test data]

(iv) analyze resulting behavior [after using invalid test data]

(J) debug and solve problems using error messages, reference materials, language documentation, and effective strategies;

- (i) debug problems using error messages
- (ii) debug problems using reference materials
- (iii) debug problems using language documentation
- (iv) debug problems using effective strategies
- (v) solve problems using error messages
- (vi) solve problems using reference materials
- (vii) solve problems using language documentation
- (viii) solve problems using effective strategies

(K) create and implement common algorithms such as finding greatest common divisor, finding the biggest number out of three, finding primes, making change, and finding the average;

- (i) create common algorithms
- (ii) implement common algorithms

(L) create program solutions that address basic error handling such as preventing division by zero and type mismatch;

- (i) create program solutions that address basic error handling

(M) select the most appropriate construct for a defined problem;

- (i) select the most appropriate construct for a defined problem

(N) create program solutions by using the arithmetic operators to create mathematical expressions, including addition, subtraction, multiplication, real division, integer division, and modulus division;

- (i) create program solutions by using the arithmetic operators to create mathematical expressions, including addition
- (ii) create program solutions by using the arithmetic operators to create mathematical expressions, including subtraction
- (iii) create program solutions by using the arithmetic operators to create mathematical expressions, including multiplication
- (iv) create program solutions by using the arithmetic operators to create mathematical expressions, including real division
- (v) create program solutions by using the arithmetic operators to create mathematical expressions, including integer division
- (vi) create program solutions by using the arithmetic operators to create mathematical expressions, including modulus division;

(O) create program solutions to problems using available mathematics library functions or operators, including absolute value, round, power, square, and square root;

- (i) create program solutions to problems using available mathematics library functions or operators, including absolute value

(ii) create program solutions to problems using available mathematics library functions or operators, including round

(iii) create program solutions to problems using available mathematics library functions or operators, including power

(iv) create program solutions to problems using available mathematics library functions or operators, including square

(v) create program solutions to problems using available mathematics library functions or operators, including square root

(P) develop program solutions that use assignment;

(i) develop program solutions that use assignment

(Q) develop sequential algorithms to solve non-branching and non-iterative problems;

(i) develop sequential algorithms to solve non-branching problems

(ii) develop sequential algorithms to solve non-iterative problems

(R) develop algorithms to decision-making problems using branching control statements;

(i) develop algorithms to decision-making problems using branching control statements

(S) develop iterative algorithms and code programs to solve practical problems;

(i) develop iterative algorithms to solve practical problems

(ii) code programs to solve practical problems

(T) demonstrate the appropriate use of the relational operators;

(i) demonstrate the appropriate use of the relational operators

(U) demonstrate the appropriate use of the logical operators; and

(i) demonstrate the appropriate use of the logical operators

(V) generate and use random numbers.

(i) generate random numbers

(ii) use random numbers

(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) discuss and explain intellectual property, privacy, sharing of information, copyright laws, and software licensing agreements;

(i) discuss intellectual property

(ii) discuss privacy

(iii) discuss sharing of information

(iv) discuss copyright laws

(v) discuss software licensing agreements

(vi) explain intellectual property

(vii) explain privacy

(viii) explain sharing of information

(ix) explain copyright laws

(x) explain software licensing agreements

(B) practice ethical acquisition and use of digital information;

(i) practice ethical acquisition of digital information

(ii) practice ethical use of digital information

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

(i) demonstrate proper digital etiquette

(ii) demonstrate responsible use of software

(iii) demonstrate knowledge of acceptable use policies

(D) investigate privacy and security measures, including strong passwords, pass phrases, and other methods of authentication and virus detection and prevention; and

(i) investigate privacy and security measures, including strong passwords

(ii) investigate privacy and security measures, including pass phrases

(iii) investigate privacy and security measures, including other methods [beyond strong passwords and pass phrases] of authentication

(iv) investigate privacy and security measures, including virus detection

(v) investigate privacy and security measures, including virus prevention

(E) investigate computing and computing-related advancements and the social and ethical ramifications of computer usage.

(i) investigate computing advancements

(ii) investigate computing-related advancements

(iii) investigate the social ramifications of computer usage.

(iv) investigate the ethical ramifications of computer usage.

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

(A) identify and describe the function of major hardware components, including primary and secondary memory, a central processing unit (CPU), and peripherals;

(i) identify the function of major hardware components, including primary memory

(ii) identify the function of major hardware components, including secondary memory

(iii) identify the function of major hardware components, including a central processing unit (CPU)

(iv) identify the function of major hardware components, including peripherals

(v) describe the function of major hardware components, including primary memory

(vi) describe the function of major hardware components, including secondary memory

(vii) describe the function of major hardware components, including a central processing unit (CPU)

(viii) describe the function of major hardware components, including peripherals

(B) differentiate between current programming languages, discuss the general purpose for each language, and demonstrate knowledge of specific programming terminology and concepts and types of software development applications;

(i) differentiate between current programming languages

(ii) discuss the general purpose for each language

(iii) demonstrate knowledge of specific programming terminology

(iv) demonstrate knowledge of specific programming concepts

(v) demonstrate knowledge of specific types of software development applications

(C) differentiate between a high-level compiled language and an interpreted language;

(i) differentiate between a high-level compiled language and an interpreted language

(D) identify and use concepts of object-oriented design;

(i) identify concepts of object-oriented design

(ii) use concepts of object-oriented design

(E) differentiate between local and global scope access variable declarations;

(i) differentiate between local and global scope access variable declarations

(F) encapsulate data and associated subroutines into an abstract data type;

(i) encapsulate data into an abstract data type

(ii) encapsulate [data-]associated subroutines into an abstract data type

(G) create subroutines that do not return values with and without the use of arguments and parameters;

(i) create subroutines that do not return values with the use of arguments

(ii) create subroutines that do not return values with the use of parameters

(iii) create subroutines that do not return values without the use of arguments

(iv) create subroutines that do not return values without the use of parameters

(H) create subroutines that return typed values with and without the use of arguments and parameters;

(i) create subroutines that return typed values with the use of arguments

(ii) create subroutines that return typed values with the use of parameters

(iii) create subroutines that return typed values without the use of arguments

(iv) create subroutines that return typed values without the use of parameters

(I) create calls to processes passing arguments that match parameters by number, type, and position;

(i) create calls to processes passing arguments that match parameters by number, type, and position

(J) compare data elements using logical and relational operators;

(i) compare data elements using logical operators

- (ii) compare data elements using relational operators

(K) identify and convert binary representation of numeric and nonnumeric data in computer systems using American Standard Code for Information Interchange (ASCII) or Unicode;

- (i) identify binary representation of numeric data in computer systems using American Standard Code for Information Interchange (ASCII) or Unicode;
- (ii) identify binary representation of nonnumeric data in computer systems using American Standard Code for Information Interchange (ASCII) or Unicode;
- (iii) convert binary representation of numeric data in computer systems using American Standard Code for Information Interchange (ASCII) or Unicode;
- (iv) convert binary representation of nonnumeric data in computer systems using American Standard Code for Information Interchange (ASCII) or Unicode;

(L) identify finite limits of numeric data such as integer wrap around and floating point precision;

- (i) identify finite limits of numeric data

(M) perform numerical conversions between the decimal and binary number systems and count in the binary number system;

- (i) perform numerical conversions between the decimal and binary number systems
- (ii) count in the binary number system

(N) choose, identify, and use the appropriate data types for integer, real, and Boolean data when writing program solutions;

- (i) choose the appropriate data types for integer data when writing program solutions
- (ii) choose the appropriate data types for real data when writing program solutions
- (iii) choose the appropriate data types for Boolean data when writing program solutions
- (iv) identify the appropriate data types for integer data when writing program solutions
- (v) identify the appropriate data types for real data when writing program solutions
- (vi) identify the appropriate data types for Boolean data when writing program solutions
- (vii) use the appropriate data types for integer data when writing program solutions
- (viii) use the appropriate data types for real data when writing program solutions
- (ix) use the appropriate data types for Boolean data when writing program solutions

(O) analyze the concept of a variable, including primitives and objects;

- (i) analyze the concept of a variable, including primitives
- (ii) analyze the concept of a variable, including objects

(P) represent and manipulate text data, including concatenation and other string functions;

- (i) represent text data, including concatenation
- (ii) represent text data, including other string functions [than concatenation]
- (iii) manipulate text data, including concatenation
- (iv) manipulate text data, including other string functions [than concatenation]

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

(i) identify the structured data type of one-dimensional arrays to traverse data

(ii) identify the structured data type of one-dimensional arrays to search data

(iii) identify the structured data type of one-dimensional arrays to modify data

(iv) use the structured data type of one-dimensional arrays to traverse data

(v) use the structured data type of one-dimensional arrays to search data

(vi) use the structured data type of one-dimensional arrays to modify data

(R) choose, identify, and use the appropriate data type or structure to properly represent the data in a program problem solution; and

(i) choose the appropriate data type or structure to properly represent the data in a program problem solution

(ii) identify the appropriate data type or structure to properly represent the data in a program problem solution

(iii) use the appropriate data type or structure to properly represent the data in a program problem solution

(S) compare strongly typed and un-typed programming languages.

(i) compare strongly typed and un-typed programming languages