2023 Mathematics *Standards of Learning*
Understanding the Standards – Computer Mathematics

This course is intended to provide students with experiences in using computer programming techniques and skills to solve problems involving mathematical models. Students enrolled in Computer Mathematics are assumed to have studied the concepts and skills in Algebra 1 and beginning Geometry.

Even though computer ideas should be introduced in the context of mathematical concepts, problem solving should be developed in the most general sense, allowing the techniques to be applied by students in many other environments. Strategies include defining the problem; developing, refining, and implementing a plan; and testing and revising the solution. Programming, ranging from simple programs involving only a few lines to complex programs involving subprograms, should permeate the entire course. Programming concepts, problem-solving strategies, and mathematical applications should be integrated throughout the course.

These standards identify fundamental principles and concepts in the field of computer science that will be used within the context of mathematical problem solving in a variety of applications. As students develop and refine skills in logic, organization, and precise expression, they will apply those skills to enhance learning in all disciplines.

Data Representation and Storage

CM.DRS.1 The student will represent data and convert data between different number systems.

Students will demonstrate the following Knowledge and Skills:

a) Represent data in different number systems, including binary, decimal, and hexadecimal.
b) Convert data between number systems (e.g., binary to decimal, decimal to hexadecimal).

CM.DRS.1 The student will represent data and convert data between different number systems. *Additional Content Background and Instructional Guidance:*

• Binary data is a type of data that only has two possible values. These values are often represented by:
  ○ Numbers:
  ▪ 0 and 1
  ○ Words:
  ▪ *Yes* and *No*
  ▪ *True* and *False*
• The decimal number system has a base of 10. It has only 10 notations, i.e., 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.
• The hexadecimal system operates with a base of 16 because there are a total of 16 notations in it: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and A, B, C, D, E, F.

CM.DRS.1 The student will represent data and convert data between different number systems.

Additional Content Background and Instructional Guidance:

- Decimal to Hexadecimal Conversion Table:

Hexadecimal Digit	Decimal Digit
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

- Number systems can be compared by how many digits (place values) it takes to represent a piece of data.
- Data in a computer program is interpreted in binary code.
- A specific number or piece of data can be represented in all other number systems by using mathematical techniques to manipulate the base number system.
- Colors can play a role in emphasizing the importance of the hexadecimal system.

CM.DRS.2 The student will differentiate between variable data types based upon their characteristics.

Students will demonstrate the following Knowledge and Skills:

a) Describe the characteristics of different variable data types, including
   i) Boolean;
   ii) character;
   iii) integer;
   iv) decimal (double/float); and
   v) string.

b) Differentiate between variable data types to determine the data type needed based upon intended use (e.g., character versus string, integer versus double/float).

CM.DRS.2 The student will differentiate between variable data types based upon their characteristics.

Additional Content Background and Instructional Guidance:

- A data type is associated with a piece of data that tells a program how to interpret its value.
- Boolean data type has only two literal constants, represented by *True* and *False*.
- A character data type can hold one letter, number, or symbol.
- Decimal is a floating decimal point type variable.
  - Represents a number using decimal numbers (0-9).
  - Uses 128 bits (28-29 significant figures) for storing and representing data.
  - Has more precision than float and double.
  - Mostly used in financial applications because of their high precision and easy to avoid rounding errors.
- Double is also a floating binary point type variable.
  - Double precision and 64 bits size (15-17 significant figures).
  - Most generally used data type for real values, except for financial applications and places where high accuracy is desired.
- Float is a floating binary point type variable.
  - Represents a number in its binary form.
  - Single precision 32 bits (6-9 significant figures) data type.
  - Used mostly in graphic libraries because of very high demand for processing power, and in conditions where rounding errors are not extremely important.
- A string is a sequence of characters and most commonly used to store text. Additionally, a string can include digits and symbols; however, when digits and symbols are included in a string, they are stored as text values.
- Treating a character as a one-letter string and using real number variables with values of 0 for *False* and 1 for *True* simulates Boolean variables.
- Describing a data type includes telling the difference in functionality and the intended use.

CM.DRS.2 The student will differentiate between variable data types based upon their characteristics.

Additional Content Background and Instructional Guidance:

- There are several types of variables, each used for different types of data. The numeric types include integer and real number. Non-numeric types include character (a single letter), string (words and other text data), and Boolean (either *True* or *False*).
- There are advantages and disadvantages to the use of each data type and the ways in which each can serve a purpose in an overall program.

CM.DRS.3 The student will represent data using appropriate data structures.

Students will demonstrate the following Knowledge and Skills:

a) Given a specific task or problem, determine the appropriate data structure (e.g., lists, arrays, objects) to represent data.
b) Perform tasks related to lists or arrays (one-dimensional or two-dimensional), including
   i) declare a list or array (one-dimensional or two-dimensional);
   ii) choose an appropriate data type for a list or an array; and
   iii) fill the list or array with data.
c) Access and manipulate a particular element of a list or an array.
d) Implement predefined objects to consolidate related information of different data types.

CM.DRS.3 The student will represent data using appropriate data structures.

Additional Content Background and Instructional Guidance:

- A list refers to an array that can change sizes (depending on the programming language used).
- An array is a data structure that serves as a collection of values or objects.
- Data structures are collections of components that are given a single name and whose organization is characterized by the method used to access the individual components.
- Two-dimensional arrays may be viewed as an array of multiple one-dimensional arrays.
- Data files in a computer program can be used as both a source of input data and as a way to save data for the next program execution.
- A predefined object is a basic data structure used to organize data of different types and referenced by name.

Components of Programming

CM.CP.1 The student will design a step-by-step plan to perform a task or solve a problem, including those arising from mathematical or interdisciplinary contexts.

Students will demonstrate the following Knowledge and Skills:

a) Design a step-by-step plan to perform a task or solve a problem using a flowchart or pseudocode that outlines the subtasks needed.

b) Define the variables needed to perform a task or solve a problem.

c) Define the constraints of a task or problem (e.g., pre-conditions, post-conditions) to determine the desired input and output.

CM.CP.1 The student will design a step-by-step plan to perform a task or solve a problem, including those arising from mathematical or interdisciplinary contexts.

Additional Content Background and Instructional Guidance:

- A step-by-step plan to perform a task or solve a problem can be in the form of a flowchart, pseudocode, or both.
    - A flowchart is a diagram of a program plan. It uses a set of shapes and arrows to display the logic flow of a program. Flowcharts are effective in displaying the flow of conditional structures and looping structures; however, as the programs increase in complexity, flowcharts tend to become cumbersome.
    - Pseudocode is a combination of English and calculator/computer commands used to plan an algorithm. Pseudocode is usually written in semi-outline form and is often little more than a list of jobs that need to be accomplished.
    - A hierarchy chart is a type of diagram of the plan for solving a problem. Hierarchy charts are good for analyzing the overarching level of complex programs, but do not handle loop structures and conditionals efficiently.
- Variables should be properly declared in their appropriate scope and named according to their function.
- Design includes determining whether the problem is solvable under given conditions.
- Pre-conditions can be built into the algorithm, such as input requirements/restrictions.
- Post-conditions can determine whether the desired result will be *true* after a function returns or at the end of a code segment.

CM.CP.2 The student will construct Boolean expressions and implement conditional statements.

Students will demonstrate the following Knowledge and Skills:

a) Write and implement Boolean expressions using logical and relational operators (e.g., !, &&, ||, ==, <, >, >=, <=, !=).
b) Write and implement "if" conditional statements.
c) Write and implement "if/else" conditional statements.
d) Write and implement compound conditional statements (e.g., nested conditionals, chained conditional statements).
e) Determine which parts of an algorithm are executed based on a condition being *true* or *false*.

CM.CP.2 The student will construct Boolean expressions and implement conditional statements.

Additional Content Background and Instructional Guidance:

- Boolean logic is a system using variables with only two values: *True* and *False*.
- Logical operators have a lower precedence than arithmetic or relational operators. Among logical operators, the AND operator has higher precedence than the OR operator.
- **Logical Operators:**

Symbol	Operation	Example	Description
&&	AND	exp1 && exp2	*True* only if both exp1 and exp2 are true; otherwise, *false*.
||	OR	exp1 || exp2	*True* if either exp1 or exp2 is true; *false* (0) only if both are false.
!	NOT	!exp1	*False* if exp1 is true; *true* if exp1 is false.

- **Relational Operators:**

Symbol	Operation	Example	Description
==	Equal	$x == y$	*True* if *x* is equal to *y*.
>	Greater than	$x > y$	*True* if *x* is greater than *y*.
<	Less than	$x < y$	*True* if *x* is less than *y*.
>=	Greater than or equal to	$x >= y$	*True* if *x* is greater than or equal to *y*.
<=	Less than or equal to	$x <= y$	*True* if *x* is less than or equal to *y*.
!=	Not equal to	$x != y$	*True* if *x* is not equal to *y*.

CM.CP.2 The student will construct Boolean expressions and implement conditional statements.

Additional Content Background and Instructional Guidance:

- The "if" statement is a fundamental control structure that allows branches in the flow of control.
- *If* is used to create a single selection control structure. The *If* command is followed by a condition which can be determined to be *true* or *false* and is also referred to as a Boolean test. This differs from a Boolean variable, which is available in many programming languages but not on most graphing calculators and stores *true* or *false* as its data. *If* the condition is *true*, control goes to the next command and it is executed. *If* the condition is *false*, the next command, and ONLY the next command, is skipped and control will go to the following command.
- *If-Then* executes a group of commands rather than a single command. The *Then* command immediately follows *If* to mark the beginning of the group and the *End* command marks the end of the group.
- The use of "*if*" statements, "*if/then/else*" statements, case statements, and Boolean logic can increase the complexity of algorithms.
- Use selection to determine which parts of an algorithm are executed based on a condition being *true* or *false*.
- Implement nested conditionals as an alternative to the use of complex Boolean expressions.
- Chained conditional refers to a sequence of *if/else* conditional statements.

CM.CP.3 The student will perform iteration with loops.

Students will demonstrate the following Knowledge and Skills:

a) Write and implement "while" and "for" loops.

b) Differentiate between loops that run a fixed number of times and loops that run an indefinite number of times (e.g., stopping dependent on variable conditions).

c) Identify conditions that cause infinite loops.

d) Determine the outcome of code segments that include loops.

CM.CP.3 The student will perform iteration with loops.

Additional Content Background and Instructional Guidance:

- Loops are control structures used to repeat a specific section of code until a condition is met.
- Loops that are infinite run indefinitely.
- The three types of loop structures are iterative, pre-test, and post-test. Each has its place in the programmer's toolbox.
    - Iterative Loops (counting loops) are the most basic type of loop and are used when the number of times a task is to be done is known or when the first and last value of a variable along with the size of the increment from first to last is known.
    - Pre-Test Loops ask the "do I do this loop" question before executing another trip through the loop. As program control reaches the top of a pre-test loop, it tests a do condition and if the answer is *Yes*, the loop is executed. If the answer is *No*, control is passed to the next line after the bottom of the loop. This positioning of the question leads to the possibility that the loop might not ever be executed.
    - Post-Test Loops are similar to the pre-test loops in that the loop variable must be initialized before the loop begins and must be updated within the body of the loop. There are two main differences between the loops. The first is the position of the decision about recycling through another iteration of the loop. In a post-test loop the question is asked after the body of the loop has been executed instead of before as in the pre-test loop. The second is in the way the "do it again" condition is written. In a pre-test (While) loop, the condition is a continuing condition (Keep doing this loop While X is less than 10). It is written so that if the result of the condition is true, the loop recycles. In a post-test loop (Repeat), the condition is an exit condition (Repeat this loop until X is greater than or equal to 10). The positioning of the recycle condition in the post-test loop guarantees that the loop will be executed at least one time.
- Loops that are controlled run for a set number of iterations.
- Controlled loops can be counter controlled or sentinel controlled.
    - Counter controlled loops have a predetermined number of iterations.

CM.CP.3 The student will perform iteration with loops.

Additional Content Background and Instructional Guidance:

  - Sentinel controlled loops have an unknown number of iterations that is determined inside the loop.

- For loops are count controlled loops.
- Nested loops are loops that include other loops.
- The introduction of two-dimensional arrays provides the opportunity to include writing and implementing nested loops.
- Tables can be used to trace code that includes loops and to determine the outcome in each iteration.
- The loop control variable should have an initial value before the loop begins.
- The programmer must be aware of the value of the loop variable before the start of the loop, because it is possible that the loop might not be executed if the value does not meet the While condition.
- The While condition is an "I want to do this loop *if* this is true" condition. If the condition is true, the loop will be executed. When the program reaches the top of the loop, it tests the condition and, if the answer is true, the body of the loop is executed. Care must be given to the condition since a faulty condition can lead to an infinite loop or to the loop never being executed. As in the For loop, the End statement marks the bottom of the loop and control is passed back to the top of the loop for testing the While condition.
- One key difference between the For-loop and the pre-test and post-test loops is the way the loop control variable is updated. In a For-loop, the value of the loop variable is automatically updated by the step parameter every time the loop reaches the bottom. In both the pre-test and post-test loop, the programmer must add a line of code that will change the value of the loop control variable.

CM.CP.4 The student will write and implement the output phase of a computer program.

Students will demonstrate the following Knowledge and Skills:

a) Write and implement the output phase of a computer program, which may include:
   i) formatting output in text-based environments;
   ii) displaying output through a graphical user interface; and
   iii) sending output to a physical device (e.g., speakers, robots, LED lights).

b) Write output to a file.

CM.CP.4 The student will write and implement the output phase of a computer program.

Additional Content Background and Instructional Guidance:

- The output is the result produced by the execution of the program.
- The output can include script, file, graphics, audio, LED lights, robotic movement or any other medium that can receive the result of the program execution.
- Designing the output phase of a computer program and displaying the results are contingent upon the format requirements of the program.
- Completely list all data to be displayed as output, including the variables that will be used to represent them in the program.
- Completely list all formulas required by the program using the variables detailed in the output and input sections.
- Before more complex challenges can be undertaken, the student must understand how to label locations in memory and get data into the memory locations.

CM.CP.5 The student will write and implement the input phase of a computer program.

Students will demonstrate the following Knowledge and Skills:

a) Write and implement input statements to store user given values into a program.
b) Validate input data using exception coding (e.g., using a "while" loop to control valid input by a user).
c) Determine what output a program will produce given a specific input.

CM.CP.5 The student will write and implement the input phase of a computer program.

Additional Content Background and Instructional Guidance:

- The input phase of a computer program is a means for the program to obtain information.
- A program requires data on which it can operate.
- The input phase can be formatted according to the specifications of the program.
- Input can be through a number of media (e.g., a file, keyboard, mouse, audio).
- Input is stored in variables and more advanced data structures such as lists, arrays and objects.
- Input can be obtained by assignment statements or by input statements with appropriate prompts.
- Input is validated using a variety of methods such as control structures.
- Input should be verified whenever possible and stored in the appropriate variable type.
- The input of a program can affect its function and its output.
- Completely list all data necessary to input into the program using assignment input statements. Include the variables that will be used to represent them in the program.
- Before more complex challenges can be undertaken, the student must understand how to label locations in memory and get data into the memory locations.

CM.CP.6 The student will implement library functions.

Students will demonstrate the following Knowledge and Skills:

a) Implement library functions to process data.
b) Implement library functions to perform mathematical operations (e.g., random, absolute value, square root, power).
c) Implement void library functions and return library functions.
d) Implement overloaded library functions.

CM.CP.6 The student will implement library functions.

Additional Content Background and Instructional Guidance:

- Functions can be user-defined or standard library functions.
- A library is a location in a computer program that includes functions.
- Library functions are built in functions placed in a common library.
- Most languages have at least one designated library that includes functions that perform mathematical operations.
- Functions that return a value are called return functions.
- Functions that do not return a value are called void functions.
- Some languages allow the use of more than one function with the same name. These functions are called Overloaded functions.
- Overloaded functions are differentiated by order, and/or type of parameters.

CM.CP.7 The student will write and implement user-defined functions.

Students will demonstrate the following Knowledge and Skills:

a) Write and implement a user-defined function to complete a task or sub-task.
b) Write and implement void functions and return functions.
c) Write and implement functions that accept parameters.

CM.CP.7 The student will write and implement user-defined functions. *Additional Content Background and Instructional Guidance:*
• User defined functions or methods are subroutines that can be defined by the user and called as needed in the body of a program to perform a specific action. • User defined functions can be return type or void type. • Return functions must return a value. • Void functions perform a task but do not return a value.

CM.CP.8 The student will implement pre-defined algorithms, including search routines and sort routines.

Students will demonstrate the following Knowledge and Skills:

a) Differentiate between types of search routines.
b) Differentiate between types of sort routines.
c) Implement pre-defined algorithms.
d) Implement a search routine on a one-dimensional list or an array, including sequential search and binary search.
e) Implement a sort routine on a one-dimensional list or an array (e.g., selection sort, insertion sort, merge sort).

CM.CP.8 The student will implement pre-defined algorithms, including search routines and sort routines.

Additional Content Background and Instructional Guidance:

- A routine is a sequence of code intended for the execution of a program.
- Search routines are sequences of code that use loops to find a given target.
- Differentiate between search routines that include linear and binary search.
- Linear search routines are sequential whereas binary search routines are logarithmic.
- Sorting algorithms (routines) use elements of an array or list as an input and arrange the elements into a meaningful order so that they can be analyzed effectively.
- Differentiate between sort algorithms (routines) that include one or more of the following:
    - selection sort;
    - bubble sort;
    - insertion sort; and,
    - merge sort.

Applications of Programming

CM.AP.1 The student will write and implement programs using sequencing, selection, and iteration to perform a specific task or solve a problem, including those arising from mathematical and interdisciplinary contexts.

Students will demonstrate the following Knowledge and Skills:

a) Determine what components of programming are needed to implement a step-by-step plan to perform a specific task or solve a problem.
b) Write a computer program that includes sequencing, selection (conditionals), and iteration (loops).
c) Write and implement computer programs to solve mathematical problems using
   i) formulas and equations;
   ii) functions;
   iii) probability and statistics; and
   iv) data-analysis.

CM.AP.1 The student will write and implement programs using sequencing, selection, and iteration to perform a specific task or solve a problem, including those arising from mathematical and interdisciplinary contexts.

Additional Content Background and Instructional Guidance:

- After defining a problem that must be solved, a first step is to determine which programming concepts and tools can be used to solve the problem.
- Sequencing involves code being run with a top-down or one line at a time approach.
- Conditional Statements:
  - Boolean statements
  - IF statements – Single selection
  - IF/ELSE statements – Double selection
  - IF/ELSE/GOTO – Multiple selection
- Selection is when a section of code is run only if a certain condition is met.
- *If* is used to create a single selection control structure. The *If* command is followed by a condition which can be determined to be *true* or *false* and is also referred to as a Boolean test. *If* the condition is *true*, control goes to the next command and it is executed. *If* the condition is *false*, the next command, and ONLY the next command, is skipped and control will go to the following command.
- *If-Then-Else* creates a double selection control structure. *If* the condition is *true*, the commands following the *then* statement are executed. *If* the condition is *false*, the commands following the *else* statement are executed.
- Iteration refers to code being repeated until a specific end result is met.
- The analysis of data in charts, graphs, and tables may be included in the design and implementation of a computer program.

CM.AP.2 The student will create documentation using written comments to annotate the intended purpose of the components of a user-created program.

Students will demonstrate the following Knowledge and Skills:

a) Create documentation using written comments to:
   i) describe the overall purpose of a program;
   ii) align a previously created step-by-step plan to a written program;
   iii) describe pre-conditions and post-conditions; and
   iv) improve the readability of a program.

CM.AP.2 The student will create documentation using written comments to annotate the intended purpose of the components of a user-created program.

Additional Content Background and Instructional Guidance:

- Program documentation provides useful information for the programmer and users of a program.
- Documentation happens in the program's design and coding.
- Using proper "comments" in code helps ensure future readability of the code.
- When creating code, the use of white space and indentation can give code an organized look.
- Using comments, a coder may be able to describe programming code more precisely.
- Pre-conditions are conditions that must be *true* for a method or function to run properly.
- Post-conditions are conditions that must be *true* after the program's method or function output.

CM.AP.3 The student will verify how programs access and process variables.

Students will demonstrate the following Knowledge and Skills:

a) Verify that the variable types are aligned to the purpose of the algorithm.
b) Verify that global variables are set to constant values before run time.
c) Differentiate between the scopes of variables (e.g., global scope versus local scope) and verify the intended use.

CM.AP.3 The student will verify how programs access and process variables.

Additional Content Background and Instructional Guidance:

- A data type specifies what type of value a variable can accept.
- Primitive data types (e.g., int, float, char, Boolean) are the most basic data types.
- Class data types (e.g., string, scanner, double) are types that are used for object-oriented programming.
- A constant variable is a variable that cannot be changed through the program's execution.
- Local variables are declared inside a function block.
- Local variables exist only within the program or subroutine in which they are written.
- Local variables scope resides only inside the function in which they were declared.
- Global variables are declared outside of a function.
- Global variables can be accessed by any program or subroutine.
- Global variables can be used inside a program's function as well as outside.
- The scope of a variable is the portion of the program in which the variable can be used.
- Students must understand that variables in programming are not the same as variables in other mathematics courses. A variable is traditionally defined in mathematics courses as a symbol that represents an unknown value. In programming, a variable is a name given to a location in memory.

CM.AP.4 The student will translate a mathematical expression or statement into computer code.

Students will demonstrate the following Knowledge and Skills:

a) Declare, initialize, and assign variables to represent mathematical expressions or statements.
b) Implement order of operations, including logical and relational operators.
c) Translate a mathematical expression or statement into a programming statement(s).

CM.AP.4 The student will translate a mathematical expression or statement into computer code. *Additional Content Background and Instructional Guidance:*
• Declaring a variable includes specifying the data type and giving the variable a name (for example: int num). • Initializing a variable assigns the variable an initial value (for example num = 5). • When evaluating statements with multiple operators, order of operations must be used. • Using the acronym PEMDAS (Parentheses, Exponents, Multiplication/Division, Addition/Subtraction) is useful in determining the order of operations. • Writing programs involving formulas provides a bridge between this course and discussions of formulas and variables that took place in previous mathematics courses. Programming will allow students to instruct the device/software to substitute the assigned values for the appropriate variables and simplify the expression. By doing this, students can program the device to accomplish the tasks that may have caused them difficulty in previous mathematics courses.

CM.AP.5 The student will trace existing code to interpret the intended purpose.

Students will demonstrate the following Knowledge and Skills:

a) Trace existing code of an algorithm to
   i) identify values at each stage of an algorithm; and
   ii) predict return values of functions given specific arguments.

b) Use tracing to describe the intended purpose of existing code for an algorithm.

CM.AP.5 The student will trace existing code to interpret the intended purpose. *Additional Content Background and Instructional Guidance:*
• Code tracing is a method in which a programmer evaluates the execution of a program or code step-by-step. • While tracing the execution of a program it is possible to track the value of a variable as it changes through the program's execution. • A trace table consists of rows and columns that can be used to test an algorithm and predict how the computer program will run the algorithm. • Tracing code is common when predicting how a computer will execute conditionals and loops.

Evaluation of Programming

CM.EP.1 The student will test a program to match a sample output, using a set of data.

Students will demonstrate the following Knowledge and Skills:

a) Produce a given output by entering a data set.
b) Test a program including boundary cases and inaccurate data types to verify the intended outcomes.

CM.EP.1 The student will test a program to match a sample output, using a set of data. *Additional Content Background and Instructional Guidance:*
• A program will produce different output results depending on the input. • The output versus input of a program can determine its functionality. • Programs might include boundary cases and/or invalid data filtering. • Programs should be tested for all cases including those for which the program was not intended. • Detail at least one complete set of input data and the correct output data. • Some programs may require more than one set for a satisfactory test.

CM.EP.2 The student will identify errors and debug a program using various techniques.

Students will demonstrate the following Knowledge and Skills:

a) Differentiate among syntax errors, runtime errors, and logic errors.
b) Debug a program using various techniques:
   i) interpret syntax and runtime error messages;
   ii) place controlled breaks;
   iii) output intermediate results;
   iv) disable a section of code by converting it into a comment;
   v) trace code to identify logic errors; and
   vi) use debugging tools available in the programming environment.

CM.EP.2 The student will identify errors and debug a program using various techniques. *Additional Content Background and Instructional Guidance:*
• Syntax errors are errors in the syntax of the code. These errors are identified by the compiler before the program is compiled or run. The compiler is a computer program that translates computer code within one programming language (source code) into another language (target language). • Runtime errors are errors that take place while executing a program. • Logic errors occur when there is a flaw in the logic or structure of a program. • Debugging is the process of identifying and removing errors from a program. • "Commenting out" is the practice of disabling a code segment by converting to a comment in order to isolate an error.

CM.EP.3 The student will compare and contrast the efficiency of computer programs.

Students will demonstrate the following Knowledge and Skills:

a) Compare and contrast the efficiency of computer programs in terms of
   i) complexity of algorithms with the same intended outcomes;
   ii) memory space used; and
   iii) run time.

CM.EP.2 The student will identify errors and debug a program using various techniques.

Additional Content Background and Instructional Guidance:

- The efficiency of computer programs can be linked to algorithmic efficiency and runtime execution speed.
- The efficiency of an algorithm relates to the number of computational resources used by the algorithm.
- The resources an algorithm mostly uses are processing power and memory.