Virginia Mathematics

Computer Mathematics

Data Representation and Storage	**DRS.1** The student will represent data and convert data between different number systems. CM.DRS.1
	DRS.2 The student will differentiate between variable data types based upon their characteristics. CM.DRS.2
	DRS.3 The student will represent data using appropriate data structures. CM.DRS.3
Represent data in different number systems, including binary, decimal, and hexadecimal. CM.DRS.1.A	**a** Represent data in different number systems, including binary, decimal, and hexadecimal. CM.DRS.1.A
Convert data between number systems (e.g., binary to decimal, decimal to hexadecimal). CM.DRS.1.B	**b** Convert data between number systems (e.g., binary to decimal, decimal to hexadecimal). CM.DRS.1.B
Describe the characteristics of different variable data types, including CM.DRS.2.A	**a** Describe the characteristics of different variable data types, including CM.DRS.2.A
Boolean; CM.DRS.2.A.I	**i** Boolean; CM.DRS.2.A.I
character; CM.DRS.2.A.II	**ii** character; CM.DRS.2.A.II
integer; CM.DRS.2.A.III	**iii** integer; CM.DRS.2.A.III
decimal (double/float); and CM.DRS.2.A.IV	**iv** decimal (double/float); and CM.DRS.2.A.IV
string. CM.DRS.2.A.V	**v** string. CM.DRS.2.A.V

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.B	**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.B
Given a specific task or problem, determine the appropriate data structure (e.g., lists, arrays, objects) to represent data. CM.DRS.3.A	**a** **Given a specific task or problem, determine the appropriate data structure (e.g., lists, arrays, objects) to represent data.** CM.DRS.3.A
Perform tasks related to lists or arrays (one-dimensional or two-dimensional), including CM.DRS.3.B	**b** **Perform tasks related to lists or arrays (one-dimensional or two-dimensional), including** CM.DRS.3.B
declare a list or array (one-dimensional or two-dimensional); CM.DRS.3.B.I	**i** **declare a list or array (one-dimensional or two-dimensional);** CM.DRS.3.B.I
choose an appropriate data type for a list or an array; and CM.DRS.3.B.II	**ii** **choose an appropriate data type for a list or an array; and** CM.DRS.3.B.II
fill the list or array with data. CM.DRS.3.B.III	**iii** **fill the list or array with data.** CM.DRS.3.B.III
Access and manipulate a particular element of a list or an array. CM.DRS.3.C	**c** **Access and manipulate a particular element of a list or an array.** CM.DRS.3.C
Implement predefined objects to consolidate related information of different data types. CM.DRS.3.D	**d** **Implement predefined objects to consolidate related information of different data types.** CM.DRS.3.D

Components of Programming	**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. CM.CP.1
	CP.2 The student will construct Boolean expressions and implement conditional statements. CM.CP.2
	CP.3 The student will perform iteration with loops. CM.CP.3
	CP.4 The student will write and implement the output phase of a computer program. CM.CP.4
	CP.5 The student will write and implement the input phase of a computer program. CM.CP.5
	CP.6 The student will implement library functions. CM.CP.6
	CP.7 The student will write and implement user-defined functions. CM.CP.7
	CP.8 The student will implement pre-defined algorithms, including search routines and sort routines. CM.CP.8
Design a step-by-step plan to perform a task or solve a problem using a flowchart or pseudocode that outlines the subtasks needed. CM.CP.1.A	**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. CM.CP.1.A
Define the variables needed to perform a task or solve a problem. CM.CP.1.B	**b** Define the variables needed to perform a task or solve a problem. CM.CP.1.B
Define the constraints of a task or problem (e.g., pre-conditions, post-conditions) to determine the desired input and output. CM.CP.1.C	**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.C
Write and implement Boolean expressions using logical and relational operators (e.g., !, &&, ||, ==, <, >, >=, <=, !=). CM.CP.2.A	**a** Write and implement Boolean expressions using logical and relational operators (e.g., !, &&, ||, ==, <, >, >=, <=, !=). CM.CP.2.A

Write and implement "if" conditional statements. CM.CP.2.B	**b** **Write and implement "if" conditional statements.** CM.CP.2.B
Write and implement "if/else" conditional statements. CM.CP.2.C	**c** **Write and implement "if/else" conditional statements.** CM.CP.2.C
Write and implement compound conditional statements (e.g., nested conditionals, chained conditional statements). CM.CP.2.D	**d** **Write and implement compound conditional statements (e.g., nested conditionals, chained conditional statements).** CM.CP.2.D
Determine which parts of an algorithm are executed based on a condition being true or false. CM.CP.2.E	**e** **Determine which parts of an algorithm are executed based on a condition being true or false.** CM.CP.2.E
Write and implement "while" and "for" loops. CM.CP.3.A	**a** **Write and implement "while" and "for" loops.** CM.CP.3.A
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). CM.CP.3.B	**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).** CM.CP.3.B
Identify conditions that cause infinite loops. CM.CP.3.C	**c** **Identify conditions that cause infinite loops.** CM.CP.3.C
Determine the outcome of code segments that include loops. CM.CP.3.D	**d** **Determine the outcome of code segments that include loops.** CM.CP.3.D
Write and implement the output phase of a computer program, which may include: CM.CP.4.A	**a** **Write and implement the output phase of a computer program, which may include:** CM.CP.4.A

formatting output in text-based environments; CM.CP.4.A.I	i formatting output in text-based environments; CM.CP.4.A.I
displaying output through a graphical user interface; and CM.CP.4.A.II	ii displaying output through a graphical user interface; and CM.CP.4.A.II
sending output to a physical device (e.g., speakers, robots, LED lights). CM.CP.4.A.III	iii sending output to a physical device (e.g., speakers, robots, LED lights). CM.CP.4.A.III
Write output to a file. CM.CP.4.B	b Write output to a file. CM.CP.4.B
Write and implement input statements to store user given values into a program. CM.CP.5.A	a Write and implement input statements to store user given values into a program. CM.CP.5.A
Validate input data using exception coding (e.g., using a “while” loop to control valid input by a user). CM.CP.5.B	b Validate input data using exception coding (e.g., using a “while” loop to control valid input by a user). CM.CP.5.B
Determine what output a program will produce given a specific input. CM.CP.5.C	c Determine what output a program will produce given a specific input. CM.CP.5.C
Implement library functions to process data. CM.CP.6.A	a Implement library functions to process data. CM.CP.6.A
Implement library functions to perform mathematical operations (e.g., random, absolute value, square root, power). CM.CP.6.B	b Implement library functions to perform mathematical operations (e.g., random, absolute value, square root, power). CM.CP.6.B

Implement void library functions and return library functions. CM.CP.6.C	**c** **Implement void library functions and return library functions.** CM.CP.6.C
Implement overloaded library functions. CM.CP.6.D	**d** **Implement overloaded library functions.** CM.CP.6.D
Write and implement a user-defined function to complete a task or sub-task. CM.CP.7.A	**a** **Write and implement a user-defined function to complete a task or sub-task.** CM.CP.7.A
Write and implement void functions and return functions. CM.CP.7.B	**b** **Write and implement void functions and return functions.** CM.CP.7.B
Write and implement functions that accept parameters. CM.CP.7.C	**c** **Write and implement functions that accept parameters.** CM.CP.7.C
Differentiate between types of search routines. CM.CP.8.A	**a** **Differentiate between types of search routines.** CM.CP.8.A
Differentiate between types of sort routines. CM.CP.8.B	**b** **Differentiate between types of sort routines.** CM.CP.8.B
Implement pre-defined algorithms. CM.CP.8.C	**c** **Implement pre-defined algorithms.** CM.CP.8.C
Implement a search routine on a one-dimensional list or an array, including sequential search and binary search. CM.CP.8.D	**d** **Implement a search routine on a one-dimensional list or an array, including sequential search and binary search.** CM.CP.8.D
Implement a sort routine on a one-dimensional list or an array (e.g., selection	**e** **Implement a sort routine on a one-dimensional list or an array (e.g., selection sort, insertion sort, merge sort).** CM.CP.8.E

sort, insertion sort, merge sort). CM.CP.8.E	
Applications of Programming	**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. CM.AP.1
	AP.2 The student will create documentation using written comments to annotate the intended purpose of the components of a user-created program. CM.AP.2
	AP.3 The student will verify how programs access and process variables. CM.AP.3
	AP.4 The student will translate a mathematical expression or statement into computer code. CM.AP.4
	AP.5 The student will trace existing code to interpret the intended purpose. CM.AP.5
Determine what components of programming are needed to implement a step-by-step plan to perform a specific task or solve a problem. CM.AP.1.A	**a** Determine what components of programming are needed to implement a step-by-step plan to perform a specific task or solve a problem. CM.AP.1.A
Write a computer program that includes sequencing, selection (conditionals), and iteration (loops). CM.AP.1.B	**b** Write a computer program that includes sequencing, selection (conditionals), and iteration (loops). CM.AP.1.B
Write and implement computer programs to solve mathematical problems using CM.AP.1.C	**c** Write and implement computer programs to solve mathematical problems using CM.AP.1.C
formulas and equations; CM.AP.1.C.I	**i** formulas and equations; CM.AP.1.C.I
functions; CM.AP.1.C.II	**ii** functions; CM.AP.1.C.II
probability and statistics; and CM.AP.1.C.III	**iii** probability and statistics; and CM.AP.1.C.III

data-analysis. CM.AP.1.C.IV	iv data-analysis. CM.AP.1.C.IV
Create documentation using written comments to: CM.AP.2.A	a Create documentation using written comments to: CM.AP.2.A
describe the overall purpose of a program; CM.AP.2.A.I	i describe the overall purpose of a program; CM.AP.2.A.I
align a previously created step-by-step plan to a written program; CM.AP.2.A.II	ii align a previously created step-by-step plan to a written program; CM.AP.2.A.II
describe pre-conditions and post-conditions; and CM.AP.2.A.III	iii describe pre-conditions and post-conditions; and CM.AP.2.A.III
improve the readability of a program. CM.AP.2.A.IV	iv improve the readability of a program. CM.AP.2.A.IV
Verify that the variable types are aligned to the purpose of the algorithm. CM.AP.3.A	a Verify that the variable types are aligned to the purpose of the algorithm. CM.AP.3.A
Verify that global variables are set to constant values before run time. CM.AP.3.B	b Verify that global variables are set to constant values before run time. CM.AP.3.B
Differentiate between the scopes of variables (e.g., global scope versus local scope) and verify the intended use. CM.AP.3.C	c Differentiate between the scopes of variables (e.g., global scope versus local scope) and verify the intended use. CM.AP.3.C
Declare, initialize, and assign variables to represent mathematical expressions or statements. CM.AP.4.A	a Declare, initialize, and assign variables to represent mathematical expressions or statements. CM.AP.4.A

Implement order of operations, including logical and relational operators. CM.AP.4.B	**b** Implement order of operations, including logical and relational operators. CM.AP.4.B
Translate a mathematical expression or statement into a programming statement(s). CM.AP.4.C	**c** Translate a mathematical expression or statement into a programming statement(s). CM.AP.4.C
Trace existing code of an algorithm to CM.AP.5.A	**a** Trace existing code of an algorithm to CM.AP.5.A
identify values at each stage of an algorithm; and CM.AP.5.A.I	**i** identify values at each stage of an algorithm; and CM.AP.5.A.I
predict return values of functions given specific arguments. CM.AP.5.A.II	**ii** predict return values of functions given specific arguments. CM.AP.5.A.II
Use tracing to describe the intended purpose of existing code for an algorithm. CM.AP.5.B	**b** Use tracing to describe the intended purpose of existing code for an algorithm. CM.AP.5.B
Evaluation of Programming	**EP.1** The student will test a program to match a sample output, using a set of data. CM.EP.1
	EP.2 The student will identify errors and debug a program using various techniques. CM.EP.2
	EP.3 The student will compare and contrast the efficiency of computer programs. CM.EP.3
Produce a given output by entering a data set. CM.EP.1.A	**a** Produce a given output by entering a data set. CM.EP.1.A
Test a program including boundary cases and inaccurate data types to verify the intended outcomes. CM.EP.1.B	**b** Test a program including boundary cases and inaccurate data types to verify the intended outcomes. CM.EP.1.B

Differentiate among syntax errors, runtime errors, and logic errors. CM.EP.2.A	**a** **Differentiate among syntax errors, runtime errors, and logic errors.** CM.EP.2.A
Debug a program using various techniques: CM.EP.2.B	**b** **Debug a program using various techniques:** CM.EP.2.B
interpret syntax and runtime error messages; CM.EP.2.B.I	**i** **interpret syntax and runtime error messages;** CM.EP.2.B.I
place controlled breaks; CM.EP.2.B.II	**ii** **place controlled breaks;** CM.EP.2.B.II
output intermediate results; CM.EP.2.B.III	**iii** **output intermediate results;** CM.EP.2.B.III
disable a section of code by converting it into a comment; CM.EP.2.B.IV	**iv** **disable a section of code by converting it into a comment;** CM.EP.2.B.IV
trace code to identify logic errors; and CM.EP.2.B.V	**v** **trace code to identify logic errors; and** CM.EP.2.B.V
use debugging tools available in the programming environment. CM.EP.2.B.VI	**vi** **use debugging tools available in the programming environment.** CM.EP.2.B.VI
Compare and contrast the efficiency of computer programs in terms of CM.EP.3.A	**a** **Compare and contrast the efficiency of computer programs in terms of** CM.EP.3.A
complexity of algorithms with the same intended outcomes; CM.EP.3.A.I	**i** **complexity of algorithms with the same intended outcomes;** CM.EP.3.A.I
memory space used; and CM.EP.3.A.II	**ii** **memory space used; and** CM.EP.3.A.II
run time. CM.EP.3.A.III	**iii** **run time.** CM.EP.3.A.III