	Division of Career Technical Education				
Prog	Programming and Software Development				
Critica	lity Survey 2025				
CONTENT	STANDARD 1.0: PROFESSIONAL ORGANIZATIONS AND LEADERSHIP				
Performan	ce Standard 1.1: Effective Leadership and Participation in Career Technical Stu	Ident			
Organizati	ons (CTSO) and Professional Associations				
1.1.1	Explore the role of professional organizations and/or associations in the				
	programming and software development industry.	1.21			
1.1.2	Define the value, role, and opportunities provided through career technical				
	student organizations.	1.11			
1.1.3	Engage in career exploration and leadership development.	1.07			
CONTENT	STANDARD 2.0: INDUSTRY PRACTICES				
Performan	ice Standard 2.1: Essential Skills				
2.1.1	Compare programming paradigms including procedural and object-oriented				
	programming.	2.50			
2.1.2	Decompose complex problems into simpler, more manageable problems.	2.92			
2.1.3	Plan structure and procedures before writing programs.	2.50			
2.1.4	Write readable code following industry practices (e.g., white space, naming				
	conventions, comments).	2.46			
2.1.5	Write syntactically correct statements.	2.65			
2.1.6	Navigate a computer file system.	2.88			
2.1.7	Reference documentation (e.g., language, library, framework) to use				
	implementation details.	2.69			
2.1.8	Write software based on customer specifications.	2.50			
Performance Standard 2.2: Project Development					
2.2.1	Compare software development lifecycles (e.g., Agile, Waterfall).	1.54			
2.2.2	Describe project scope and scope creep.	1.73			
2.2.3	Initialize or clone a repository, using source control (e.g., git).	2.58			
2.2.4	Commit and push code, using source control (e.g., git).	2.65			
2.2.5	Pull code, using source control (e.g., git).	2.65			
CONTENT	STANDARD 3.0: DATA				
Performan	ce Standard 3.1: Variables and Data Types				
3.1.1	Identify the scope of a given variable.	2.68			
3.1.2	Identify the value of a variable at a given point.	2.60			
3.1.3	Declare and instantiate variables.	2.88			
3.1.4	Reassign a variable.	2.84			
3.1.5	Write code that uses primitive data types (e.g., integer, floating points,				
	boolean, character).	2.84			
3.1.6	Write code that uses reference types (e.g., string, object, array).	2.84			
3.1.7	Write code that uses operators (e.g., +, -, *, /, %).	2.88			

3.1.9 Compare primitive types and derived/reference types. 2.24 3.1.10 Define constants and enumerations. 2.40 Performance Standard 3.2: Arrays 2.80 3.2.1 Declare an array and assign values to array elements. 2.76 3.2.2 Access data stored in array elements. 2.76 3.2.3 Iterate through all elements in an array. 2.76 3.2.4 Search an array, using a loop. 2.68 3.2.5 Create multidimensional arrays. 2.08 3.2.6 Sort elements in an array. 2.20 CONTENT STANDARD 4.0: CONTROL FLOW Performance Standard 4.1: Branching and Logic 4.1.1 Execute decisions in a program, using 'f', "else-if," and "else" statements. 2.96 4.1.1 Execute decisions in a program, using a nested IF statement. 2.58 4.1.4 Execute decisions in a program, using a nested IF statement. 2.58 4.1.5 Execute decisions in a program, using a nested IF statement. 2.64 4.2.1 Create loops, using the for statement. 2.87 4.2.2 Create loops, using the for statement. 2.87 4.2.3 Write code that uses accumulators (e.g., running total, collection).					
3.1.10Define constants and enumerations.2.40Performance Standard 3.2: Arrays2.203.2.1Declare an array and assign values to array elements.2.763.2.2Access data stored in array elements.2.763.2.3Iterate through all elements in an array.2.763.2.4Search an array, using a loop.2.683.2.5Create multidimensional arrays.2.003.2.6Sort elements in an array.2.20CONTENT STANDARD 4.0: CONTROL FLOWPerformance Standard 4.1: Branching and Logic4.1.1Execute decisions in a program, using "if," "else-if," and "else" statements.2.964.1.3Create compound conditional statements with logical operators (e.g., ! [NOT], && (NDI) [OR]).2.834.1.4Execute decisions in a program, using a nested IF statement.2.04Performance Standard 4.2: Loops4.2.1Create loops, using the while statement.2.04Performance Standard 4.2: Loops4.2.2Create loops, using the or statement.2.874.2.3Write code that uses nested loops.2.264.3.1Describe reasons for writing functions (e.g., running total, collection).2.35Performance Standard 4.3: Functions3.004.3.2Write functions that require one or more parameters.2.914.3.3Write functions that require one or more parameters.2.914.3.4Write functions.2.904.3.5Call functions.3.004.3.6Pase parameters to functions.2.96	3.1.8	Cast data types.	2.36		
Performance Standard 3.2: Arrays3.2.1Declare an array and assign values to array elements.2.803.2.2Access data stored in array elements.2.763.2.3Iterate through all elements in an array.2.763.2.4Search an array, using a loop.2.683.2.5Create multidimensional arrays.2.003.2.6Sort elements in an array.2.00CONTENT STANDARD 4.0: CONTROL FLOWPerformance Standard 4.1: Branching and Logic4.1.1Execute decisions in a program, using "if," "else-if," and "else" statements.2.964.1.2Compare values with conditional operators (i.e., >, <, >=, <=, ==, !=).	3.1.9	Compare primitive types and derived/reference types.	2.24		
3.2.1 Declare an array and assign values to array elements. 2.80 3.2.2 Access data stored in array elements. 2.76 3.2.3 Iterate through all elements in an array. 2.76 3.2.4 Search an array, using a loop. 2.68 3.2.5 Create multidimensional arrays. 2.08 3.2.6 Sort elements in an array. 2.20 CONTENT STANDARD 4.0: CONTROL FLOW Performance Standard 4.1: Branching and Logic 4.1.1 Execute decisions in a program, using "if," "else-if," and "else" statements. 2.96 4.1.2 Compare values with conditional operators (i.e., -, <, >=, <=, ==, !=).	3.1.10	Define constants and enumerations.	2.40		
3.2.2 Access data stored in array elements. 2.76 3.2.3 Iterate through all elements in an array. 2.76 3.2.4 Search an array, using a loop. 2.68 3.2.5 Create multidimensional array. 2.00 3.2.6 Sort elements in an array. 2.00 CONTENT STANDARD 4.0: CONTROL FLOW Performance Standard 4.1: Branching and Logic 4.1.1 Execute decisions in a program, using "if," "else-if," and "else" statements. 2.96 4.1.3 Create compound conditional operators (i.e., >, <, >=, <=, ==, =).	Performance Standard 3.2: Arrays				
3.2.3 Iterate through all elements in an array. 2.76 3.2.4 Search an array, using a loop. 2.68 3.2.5 Create multidimensional arrays. 2.08 3.2.6 Sort elements in an array. 2.08 3.2.6 Sort elements in an array. 2.00 CONTENT STANDARD 4.0.CONTROL FLOW Performance Standard 4.1: Branching and Logic 2.20 4.1.1 Execute decisions in a program, using "if," "else-if," and "else" statements. 2.96 4.1.2 Compare values with conditional operators (i.e., >, <, >=<, <=, ==, !=).	3.2.1	Declare an array and assign values to array elements.	2.80		
3.2.4 Search an array, using a loop. 2.68 3.2.5 Create multidimensional arrays. 2.08 3.2.6 Sort elements in an array. 2.20 CONTENT STANDARD 4.0: CONTROL FLOW Performance Standard 4.1: Branching and Logic 2.11 4.1.1 Execute decisions in a program, using "if," "else-if," and "else" statements. 2.96 4.1.2 Compare values with conditional operators (i.e., >, <, >, <, >=, <=, ==, =).	3.2.2	Access data stored in array elements.	2.76		
3.2.5 Create multidimensional arrays. 2.08 3.2.6 Sort elements in an array. 2.20 CONTENT STANDARD 4.0: CONTROL FLOW Performance Standard 4.1: Branching and Logic 4.1.1 Execute decisions in a program, using "if," "else-if," and "else" statements. 2.96 4.1.1 Execute decisions in a program, using "if," "else-if," and "else" statements. 2.96 4.1.3 Create compound conditional operators (i.e., >, <, >=, <=, ==, !=).	3.2.3	Iterate through all elements in an array.	2.76		
3.2.6 Sort elements in an array. 2.20 CONTENT STANDARD 4.0: CONTROL FLOW Performance Standard 4.1: Branching and Logic 4.1.1 Execute decisions in a program, using "if," "else-if," and "else" statements. 2.96 4.1.2 Compare values with conditional operators (i.e., > <, >=, <=, ==, !=).	3.2.4	Search an array, using a loop.	2.68		
CONTENT STANDARD 4.0: CONTROL FLOW Performance Standard 4.1: Branching and Logic 4.1.1 Execute decisions in a program, using "if," "else-if," and "else" statements. 2.96 4.1.2 Compare values with conditional operators (i.e., >, <, >=, <=, ==, !=).	3.2.5	Create multidimensional arrays.	2.08		
Performance Standard 4.1: Branching and Logic 2.96 4.1.1 Execute decisions in a program, using "if," "else-if," and "else" statements. 2.96 4.1.2 Compare values with conditional operators (i.e., >, <, >=, <=, ==, !=).	3.2.6	Sort elements in an array.	2.20		
4.1.1 Execute decisions in a program, using "if," "else-if," and "else" statements. 2.96 4.1.2 Compare values with conditional operators (i.e., >, <, >=, <=, ==, 1=).	CONTENT	STANDARD 4.0: CONTROL FLOW			
4.1.2 Compare values with conditional operators (i.e., >, <, >=, <=, !=, !=).	Performan	ce Standard 4.1: Branching and Logic			
4.1.3 Create compound conditional statements with logical operators (e.g., ! [NOT], && [AND], [OR]). 2.83 4.1.4 Execute decisions in a program, using a nested IF statement. 2.58 4.1.5 Execute decisions in a program, using the switch statement. 2.04 Performance Standard 4.2: Loops 4.2.1 Create loops, using the while statement. 2.70 4.2.1 Create loops, using the for statement. 2.87 4.2.3 Write code that uses nested loops. 2.26 4.2.4 Write code that uses nested loops. 2.26 4.2.4 Write code that uses accumulators (e.g., running total, collection). 2.35 Performance Standard 4.3: Functions 4.3.1 Describe reasons for writing functions (e.g., to improve readability, reusability, maintainability). 2.61 4.3.2 Write functions with no parameters and no return value. 2.52 4.3.3 Write functions that require one or more parameters. 2.91 4.3.4 Write functions. 3.00 4.3.5 Call functions. 2.96 4.3.7 Write code that uses return values from functions. 2.91 4.3.8 Import libraries. 2.74 4.3.9 Write a recursive fun	4.1.1	Execute decisions in a program, using "if," "else-if," and "else" statements.	2.96		
&& [AND], [OR]).2.834.1.4Execute decisions in a program, using a nested IF statement.2.584.1.5Execute decisions in a program, using the switch statement.2.04Performance Standard 4.2: Loops4.2.1Create loops, using the while statement.2.704.2.2Create loops, using the for statement.2.874.2.3Write code that uses nested loops.2.264.2.4Write code that uses accumulators (e.g., running total, collection).2.35Performance Standard 4.3: Functions4.3.14.3.1Describe reasons for writing functions (e.g., to improve readability, reusability, maintainability).2.614.3.2Write functions with no parameters and no return value.2.524.3.3Write functions that require one or more parameters.2.914.3.4Write functions that return a value.2.914.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.744.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.57<	4.1.2	Compare values with conditional operators (i.e., >, <, >=,<=, ==, !=).	2.96		
4.1.4Execute decisions in a program, using a nested IF statement.2.584.1.5Execute decisions in a program, using the switch statement.2.04Performance Standard 4.2: Loops4.2.1Create loops, using the while statement.2.704.2.2Create loops, using the for statement.2.874.2.3Write code that uses nested loops.2.264.2.4Write code that uses nested loops.2.35Performance Standard 4.3: Functions4.3.1Describe reasons for writing functions (e.g., to improve readability, reusability, maintainability).2.614.3.2Write functions with no parameters and no return value.2.524.3.3Write functions that require one or more parameters.2.914.3.4Write functions that return a value.2.914.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.744.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONS2.875.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.1.3	Create compound conditional statements with logical operators (e.g., ! [NOT],			
4.1.5Execute decisions in a program, using the switch statement.2.04Performance Standard 4.2: Loops4.2.1Create loops, using the while statement.2.704.2.2Create loops, using the for statement.2.874.2.3Write code that uses nested loops.2.264.2.4Write code that uses accumulators (e.g., running total, collection).2.35Performance Standard 4.3: Functions4.3.1A.3.1Describe reasons for writing functions (e.g., to improve readability, reusability, maintainability).2.614.3.2Write functions with no parameters and no return value.2.524.3.3Write functions that require one or more parameters.2.914.3.4Write functions that return a value.2.914.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.744.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONS2.875.1.1Write a program that produces intended output.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.00		&& [AND], [OR]).	2.83		
Performance Standard 4.2: Loops 4.2.1 Create loops, using the while statement. 2.70 4.2.2 Create loops, using the for statement. 2.87 4.2.3 Write code that uses nested loops. 2.26 4.2.4 Write code that uses accumulators (e.g., running total, collection). 2.35 Performance Standard 4.3: Functions 4.3.1 Describe reasons for writing functions (e.g., to improve readability, reusability, maintainability). 2.61 4.3.2 Write functions with no parameters and no return value. 2.52 4.3.3 Write functions that require one or more parameters. 2.91 4.3.4 Write functions that return a value. 2.91 4.3.5 Call functions. 3.00 4.3.6 Pass parameters to functions 2.96 4.3.7 Write code that uses return values from functions. 2.91 4.3.8 Import libraries. 2.74 4.3.9 Write a recursive function. 2.00 CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONS 2.87 5.1.1 Write a program that produces intended output. 2.87 5.1.2 Provide appropriate prompts for user input. 2.13	4.1.4	Execute decisions in a program, using a nested IF statement.	2.58		
4.2.1Create loops, using the while statement.2.704.2.2Create loops, using the for statement.2.874.2.3Write code that uses nested loops.2.264.2.4Write code that uses accumulators (e.g., running total, collection).2.35Performance Standard 4.3: Functions2.614.3.1Describe reasons for writing functions (e.g., to improve readability, reusability, maintainability).2.614.3.2Write functions with no parameters and no return value.2.524.3.3Write functions that require one or more parameters.2.914.3.4Write functions that return a value.2.914.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.914.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.1.5	Execute decisions in a program, using the switch statement.	2.04		
4.2.2Create loops, using the for statement.2.874.2.3Write code that uses nested loops.2.264.2.4Write code that uses accumulators (e.g., running total, collection).2.35Performance Standard 4.3: Functionsasian and the performance Standard 4.3: Functions (e.g., to improve readability, reusability, maintainability).2.614.3.1Describe reasons for writing functions (e.g., to improve readability, reusability, maintainability).2.614.3.2Write functions with no parameters and no return value.2.524.3.3Write functions that require one or more parameters.2.914.3.4Write functions that require one or more parameters.2.914.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.744.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	Performan	ce Standard 4.2: Loops			
4.2.3Write code that uses nested loops.2.264.2.4Write code that uses accumulators (e.g., running total, collection).2.35Performance Standard 4.3: Functions2.354.3.1Describe reasons for writing functions (e.g., to improve readability, reusability, maintainability).2.614.3.2Write functions with no parameters and no return value.2.524.3.3Write functions that require one or more parameters.2.914.3.4Write functions that require one or more parameters.2.914.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.744.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.2.1	Create loops, using the while statement.	2.70		
4.2.4Write code that uses accumulators (e.g., running total, collection).2.35PerformanceStandard 4.3: Functions2.614.3.1Describe reasons for writing functions (e.g., to improve readability, reusability, maintainability).2.614.3.2Write functions with no parameters and no return value.2.524.3.3Write functions that require one or more parameters.2.914.3.4Write functions that return a value.2.914.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.914.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformanceStandard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.2.2	Create loops, using the for statement.	2.87		
Performance Standard 4.3: Functions4.3.1Describe reasons for writing functions (e.g., to improve readability, reusability, maintainability).2.614.3.2Write functions with no parameters and no return value.2.524.3.3Write functions that require one or more parameters.2.914.3.4Write functions that return a value.2.914.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.744.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.2.3	Write code that uses nested loops.	2.26		
4.3.1Describe reasons for writing functions (e.g., to improve readability, reusability, maintainability).2.614.3.2Write functions with no parameters and no return value.2.524.3.3Write functions that require one or more parameters.2.914.3.4Write functions that return a value.2.914.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.914.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.2.4	Write code that uses accumulators (e.g., running total, collection).	2.35		
maintainability).2.614.3.2Write functions with no parameters and no return value.2.524.3.3Write functions that require one or more parameters.2.914.3.4Write functions that return a value.2.914.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.914.3.9Write a recursive function.2.91CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	Performan	ce Standard 4.3: Functions			
4.3.2Write functions with no parameters and no return value.2.524.3.3Write functions that require one or more parameters.2.914.3.4Write functions that return a value.2.914.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.744.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.135.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.3.1	Describe reasons for writing functions (e.g., to improve readability, reusability,			
4.3.3Write functions that require one or more parameters.2.914.3.4Write functions that return a value.2.914.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.744.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04		maintainability).	2.61		
4.3.4Write functions that return a value.2.914.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.744.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.3.2	Write functions with no parameters and no return value.	2.52		
4.3.5Call functions.3.004.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.744.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.3.3	Write functions that require one or more parameters.	2.91		
4.3.6Pass parameters to functions2.964.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.744.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.3.4	Write functions that return a value.	2.91		
4.3.7Write code that uses return values from functions.2.914.3.8Import libraries.2.744.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.3.5	Call functions.	3.00		
4.3.8Import libraries.2.744.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.3.6	Pass parameters to functions	2.96		
4.3.9Write a recursive function.2.00CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.3.7	Write code that uses return values from functions.	2.91		
CONTENT STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONSPerformance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.5.1.2Provide appropriate prompts for user input.5.1.3Take input from a user.5.1.4Take input from a file.5.1.5Validate input.5.1.6Write to a file.	4.3.8	Import libraries.	2.74		
Performance Standard 5.1: Input and Output5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	4.3.9	Write a recursive function.	2.00		
5.1.1Write a program that produces intended output.2.875.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	CONTENT	STANDARD 5.0: INPUT, DEBUGGING, AND EXCEPTIONS			
5.1.2Provide appropriate prompts for user input.2.135.1.3Take input from a user.2.265.1.4Take input from a file.2.175.1.5Validate input.2.575.1.6Write to a file.2.04	Performan	ce Standard 5.1: Input and Output			
5.1.3 Take input from a user. 2.26 5.1.4 Take input from a file. 2.17 5.1.5 Validate input. 2.57 5.1.6 Write to a file. 2.04	5.1.1	Write a program that produces intended output.	2.87		
5.1.4 Take input from a file. 2.17 5.1.5 Validate input. 2.57 5.1.6 Write to a file. 2.04	5.1.2	Provide appropriate prompts for user input.	2.13		
5.1.5 Validate input. 2.57 5.1.6 Write to a file. 2.04	5.1.3	Take input from a user.	2.26		
5.1.6 Write to a file. 2.04	5.1.4	Take input from a file.	2.17		
	5.1.5	Validate input.	2.57		
Performance Standard 5.2: Debugging	5.1.6	Write to a file.	2.04		
	Performan	ce Standard 5.2: Debugging			

5.2.1	Debug programs by printing values to the console.	2.57		
5.2.2	Inspect program state at runtime, using a debugger (e.g., breakpoints,			
	stepping through code).	2.22		
5.2.3	Inspect variable values during runtime, using a debugger.	2.17		
5.2.4	Identify the contents of the call stack.	1.91		
5.2.5	Fix syntax and logic errors.	2.87		
5.2.6	Test applications, using varied input.	2.48		
Performan	ce Standard 5.3: Exception Handling			
5.3.1	Catch exceptions.	2.26		
5.3.2	Write code that uses the finally block.	1.87		
5.3.3	Throw exceptions.	2.22		
CONTENT STANDARD 6.0: OBJECT-ORIENTED PROGRAMMING				
Performan	ce Standard 6.1: Classes and Objects			
6.1.1	Define abstraction.	2.17		
6.1.2	Describe object-oriented programming.	2.26		
6.1.3	Create classes and instantiate objects from those classes.	2.48		
6.1.4	Create properties.	2.48		
6.1.5	Write constructors.	2.52		
6.1.6	Describe public and private access (e.g., variables, methods).	2.43		
6.1.7	Overload methods and constructors.	1.96		
6.1.8	Reference the current object instance inside a class method (e.g., "this" in			
	Java).	2.30		
6.1.9	Demonstrate inheritance ("is a" relationships) by extending classes.	2.22		
6.1.10	Demonstrate composition ("has a" relationships) by using a class object as a			
	property in another class.	2.22		
6.1.11	Demonstrate polymorphism by overriding parent class methods.	2.09		
6.1.12	Implement interfaces.	2.17		
Performance Standard 6.2: Events				
6.2.1	Define and apply event handling.	2.26		
6.2.2	Handle control component events.	2.00		
6.2.3	Handle mouse and keyboard events.	1.83		