Information for Candidates

Test Format

The MathWorks Certified MATLAB Professional (MCMP) exam consists of two sections: 25 multiple-choice questions and 8 performance-based problems. MATLAB access is not permitted during the multiple-choice section of the exam. The performance-based problem section requires code segments to be written in MATLAB. MATLAB and the documentation will be available during this portion of the exam, though no other resources, online or otherwise, are permitted. To earn the MCMP credential, submissions for both sections of the exam must meet or exceed the passing criteria for the exam instance.

Writing MATLAB Code

The performance-based problems require code submissions written in MATLAB. Submissions must meet all the requirements outlined in the problem statement as well as the basic expectations outlined in the next section.

While there are always opportunities to improve upon submissions by adding additional error checking, comments, or code for edge cases, these additions need to be balanced with the time constraint of the exam. Consider moving on to other problems if spending more than 15-20 minutes on a problem. There will be no bonus points for solutions that go above and beyond the requirements. Additionally, there are no bonus points for "clever tricks" or obscure syntax. Code submissions should clearly communicate the solution to other MATLAB programmers.

Comments in the MATLAB code are welcome and appreciated to help explain the intent of the code. However, given the time constraints of the exam, comments are not required.

Expectations for Submissions

Each submission must meet minimum criteria to receive credit. The scoring process also evaluates requirements set forth in the problems statement. The table below outlines the minimum criteria:

Category	Criteria
Meets Requirements	Solutions must not: • Make system calls using `system` command, `!` operator, or any other method of accessing a system command prompt. • Use MEX-files or Simulink blocks.

	• Make calls through external interfaces to any other programming environments such as Java, Python, .NET, or ActiveX. • Make calls to undocumented functionality, or anything that does not contain explicit instructions in the documentation for use. • **Exception:** Calls to any documented, pre-existing MATLAB functions that may make use of any of the functionality outlined above are allowed.
Correct Answer/Stability	Solutions must not: • Produce run-time errors as a result of default execution as outlined in the problem statement. • Produce warnings that indicate final results are incorrect, incorrect functions are being called, or the correct functions are being called incorrectly. • **Exception:** Errors are acceptable when a problem statement explicitly requires an error for a given set of inputs or conditions.
Implementation	Solutions must not: • Use functions which indirectly change the workspace such as `assignin`, `evalin`, `eval`, and `feval`. • Write new functions or code that replicate existing MATLAB functionality (see table). • Contain Code Analyzer warnings if there is an automatic fix or a fix with instructions provided. • Violate any of the stated Vectorization Rules (see table). • Use variable names that collide with common MATLAB functions (see list of common MATLAB functions). • Contain code that grows the size of an array incrementally in a loop when the final array size is known. • **Exception:** Automatically generated code may contain Code Analyzer messages. These messages do not need to be addressed.

Vectorization Rules

Unless otherwise noted in a problem statement, the vectorization rules outlined in the table below serve as the minimum criteria for all submissions.

Rule	Accepted Application	Example Violation

Use element-wise operators to perform mathematical, relational, or logical operations on corresponding elements of arrays.	`x = rand(1, 10);`
	`y = rand(1, 10);`
	`z = x .* y;`
		`x = rand(1, 10);`
		`y = rand(1, 10);`
		`for i = 1:10`
		`   z(i) = x(i) * y(i);`
		`end`

Pass entire arrays to functions that accept them instead of passing smaller subsets individually in a loop.	`x = 1:10;`
	`y = sin(x);`
		`x = 1:10;`
		`for i = 1:10`
		`   y(i) = sin(x(i));`
		`end`

Call functions that return entire arrays in a single function call rather than building an array incrementally.	`x = rand(1, 10)`	`for i = 1:10`
		`   x(i) = rand();`
		`end`

Use vectors for extracting multiple elements of an array when indexing.	`x = rand(5);`
	`y1 = x(:, 4);`
		`x = rand(5);`
		`for i = 1:5`
		`   y1(i) = x(i, 4);`
		`end`

Use logical indexing for the extraction of elements of an array based on a condition.	`x = randn(1, 30);`
	`y = x(x > 0);`
		`x = randn(1, 30);`
		`for i = 1:30`
		`   if x(i) > 0`
		`      y = [y x(i)];`
		`   end`
		`end`

MATLAB Functionality to Know

Familiarity with the MATLAB operators, keywords, and functions in the table below is assumed knowledge for the MCMP exam. Submissions for exam problems must not recreate any of this functionality when the appropriate function already exists to address the need. Care should also be taken not to choose variable names that take precedence over these function names. Submissions for exam problems may use any other documented functions not appearing in the table, as long as it is not part of an add-on product (toolbox). Additionally, exam problems may introduce other functions as part of the problem statement.

Mathematical Operators	`+`
	`-`
	`*`
		`\`
		`^`
		`.*`
			`.\`
			`.^`

	/	./	
Mathematical Functions	sin cos tan asin acos atan abs	exp log log10 log2 nthroot round sqrt	polyfit polyval pi ceil floor mod
Array Creation Functions	ones zeros rand randi randn	true false eye linspace logspace	: (colon operator) meshgrid
Statistical Functions	sum prod cumsum cumprod mean	median min max diff	std var cov fft
Array Dimensions	length	numel	size
Set Operations	union intersect unique	sort sortrows	setdiff ismember
String Operations	strcmp strrep	strfind deblank	lower upper
Dates and Time	datenum datevec	datestr now	clock
Plotting Functions	plot plotyy loglog semilogx semilogy scatter contour surf image imagesc	pie bar hist subplot xlabel ylabel title legend	text axis ylim xlim grid hold colormap colorbar datetick
Graphics and UI Components	get set findobj findall gcf gca	uicontrol uitable uipanel uimenu uitoolbar guidata	figure axes uigetfile uiputfile msgbox close
Logical and Relational Operators	> < >=	<= == ~=	~ & |
Logical Functions	any all nnz find isequal	isa isnan isinf isempty isnumeric	isvector iscell ischar isstruct ishandle

File I/O	`load`
	`save`
	`fopen`
	`fclose`
	`fscanf`
		`fprintf`
		`disp`
		`textscan`
		`fgetl`
		`imread`
			`imwrite`
			`xlsread`
			`xlswrite`
			`dlmread`
			`dlmwrite`

Conversion Functions	`num2str`
	`str2double`
	`cell2mat`
		`num2cell`
		`mat2cell`
		`cellstr`
			`struct2cell`
			`cell2struct`
			`char`
			`logical`

Programming Keywords	`break`
	`case`
	`catch`
	`classdef`
	`continue`
	`else`
		`elseif`
		`end`
		`for`
		`function`
		`if`
			`otherwise`
			`return`
			`switch`
			`try`
			`while`

Vectorization	`repmat`
	`reshape`
	`cellfun`
		`arrayfun`
		`structfun`
			`bsxfun`
			`accumarray`

Help and Troubleshooting	`doc`
	`help`
	`whos`
	`which`
		`ver`
		`tic`
		`toc`
		`clear`
			`clc`
			`error`
			`warning`