Armstrong State University
Engineering Studies
MATLAB Marina – Functions Exercises

1. How does one call a MATLAB function? What must the function call provide to the function? What is generally done with the result returned from the function?
2. Complete the MATLAB program of Figure 1 that will call the built in MATLAB trigonometric functions `cos`, `sin`, and `tan` for an angle of $\frac{\pi}{4}$ radians and display the results of the operations. What do MATLAB's trigonometric functions return for scalar arguments?

```
% angle in radians
angle = pi/4;

% call trigonometric functions

% display results
```

Figure 1, MATLAB Program to Evaluate Built in Trigonometric Functions for Scalar Arguments

3. Complete the MATLAB program of Figure 2 that will call the built in MATLAB trigonometric functions `cos`, `sin`, and `tan` for a vector of the angles 0 radians, $\frac{\pi}{4}$ radians, and $\frac{\pi}{2}$ radians and display the results of the operations. What do MATLAB's trigonometric functions return for vector arguments?

```
% angles in radians
angles = [0, pi/4, pi/2];

% call trigonometric functions

% display results
```

Figure 2, MATLAB Program to Evaluate Built in Trigonometric Functions for Vector Argument

4. Complete the MATLAB function of Figure 3 named `evalCubic` that will evaluate a generic cubic polynomial function $g(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ for a scalar value of x. The argument `ak` should be a vector of four coefficients $[a_3, a_2, a_1, a_0]$; `ak(1)` will hold the polynomial coefficient a_3, `ak(2)` will hold the polynomial coefficient a_2, `ak(3)` will hold the polynomial coefficient a_1, and `ak(4)` will hold the polynomial coefficient a_0. The argument `x` and return parameter `gg` should be scalars.

```
function gg = evalCubic(ak, x)
% ------------------------------------------------------
% evalCubic.m
% ------------------------------------------------------
% evalCubic evaluates the cubic polynomial function
% f(x) = a3*x^3 + a2*x^2 + a1*x + a0
% ------------------------------------------------------
% usage: gg = evalCubic(ak, x)
%  ak = 1 by 4 vector of polynomial coefficients
%  x = scalar real number
%  gg = cubic polynomial evaluated for x
% ------------------------------------------------------
```

Figure 3, evalCubic Function

5. Determine an appropriate set of test cases to test the `evalCubic` function from Exercise 4. Write a test program to verify the correct operation of `evalCubic` function for the determined test cases.
6. Modify your `evalCubic` function from Exercise 4 so that it will work for a vector argument `x`. The function definition will not need to be altered but the argument `x` will now be a vector of real numbers and the function should return a result of the same length as the argument `x`. Hint: you should only need to modify the evaluation of the function to use array operations where appropriate.
7. Write a test program to call the `evalCubic` function from Exercise 6 to verify the correct operation of the function for the test cases: $ak = [0, 2, -1, 5]$ and $x = 1$ and $ak = [0, 2, -1, 5]$ and $x = [-5 : 0.1 : 5]$. Note: this is not a full set of test cases.
8. Modify your `evalCubic` function from Exercise 6 so that if an empty vector for either the `ak` or `x` parameters is passed to the function, the function will display an error message will be displayed and the function will return an empty vector as the result. Hint: this can be done using inline error handling.
9. Write a test program to call the `evalCubic` function from Exercise 8 to verify the correct operation of the function for the test cases: $ak = [0, 2, -1, 5]$ and $x = 1$, $ak = [0, 2, -1, 5]$ and $x = [-5 : 0.1 : 5]$, and $ak = [0, 2, -1, 5]$ and $x = [\ ]$. Note: this is not a full set of test cases.
10. Write a program to plot the function $g(x) = x^3 - 3.75x^2 + 4.2x + 8.7$ for the range $x = [-10 : 0.1 : 10]$. Use the `evalCubic` function from Exercise 8 to evaluate the polynomial function.
11. Determine an appropriate set of test cases and write a test program for the `incrementByN` MATLAB function of Figure 4. Verify that the `incrementByN` function operates correctly.

```
function result = incrementByN(value, n)
% ------------------------------------------------------
% incrementByN.m
% ------------------------------------------------------
% incrementByN returns the value incremented by n
% ------------------------------------------------------
% usage: result = incrementByN(value, n)
%  value = value to increment
%  n = amount to increment by
%  result = value incremented by n
% ------------------------------------------------------
% Notes: if n is not supplied the default n is 1
% ------------------------------------------------------
if (nargin < 2)
    n = 1;
end
result = value + n;
end
```

Figure 4, incrementByN Function

12. What determines the number of results that a MATLAB function returns to the calling statement?
13. What order (left to right or right to left) are function arguments matched for a function call? Which function arguments can be set of as optional arguments?
14. Write a MATLAB function `polarToRect` that will convert two-dimensional polar coordinates to rectangular coordinates. Use the following function below as a starting point.
    ```
    function [x, y] = polarToRect(r, theta)
    ```
15. Modify your `polarToRect` function of Exercise 14 so that it can operate on vectors of coordinates. In other words, the `polarToRect` function should take two vectors `r` and `theta` of the polar coordinates and return two vectors `x` and `y` of the corresponding rectangular coordinates. The function definition will be the same as in Exercise 14.
16. Write a test program that verifies the correct operation of your `polarToRect` function from Exercise 15 for the test cases: r = 1 and theta = $\pi/4$ radians and r = [1, 3, 1] and theta = [$\pi/4$, $3\pi/2$, 0] radians.
17. Modify your `polarToRect` function from Exercise 15 so that the function ensures the two vectors `r` and `theta` are the same length. If the vectors have different lengths, an error message should be displayed and an empty set should be returned for both `x` and `y`. Test the modified `polarToRect` function using the same test program as used in Exercise 16 but add a test case to test the error checking. Hint: enclose the function body of the previous `polarToRect` function with a conditional structure that will ensure that the vectors have the same before performing the conversion.
18. What does a programmer need to determine before writing the function definition?

19. Why is a proper comment header necessary for user created functions?
20. How does one test a function to verify that it operates according to the specifications?
21. Write a MATLAB function `sumPartial` that will take a vector and two indices and sum the vector elements between and including the two indices. Use the `sumVector` function of Figure 5 as a starting point. Hints: first determine the function arguments and return variables then determine what in the `sumVector` function needs to be modified for the new function. Invoking the function `sumpartial` with the vector `[3, -1, 2, 4, 8, -1, 0, 7]` and indices of 2 and 5 should return a sum of `-1 + 2 + 4 + 8 = 13`.

```
function result = sumVector(x)
% ------------------------------------------------------
% sumVector.m
% ------------------------------------------------------
% sumVector sums the elements of a vector
% ------------------------------------------------------
% Syntax: result = sumVector(x)
%  x is the vector of numbers
%  result is the sum of the elements in the vector
% ------------------------------------------------------
% Examples: r = sumVector([6, 3, 8, 12]); results in r = 29
% ------------------------------------------------------

result = 0.0;
for k = 1:1:length(x)
     result = result + x(k);
end
end
```

Figure 5, sumVector Function

22. Determine an appropriate set of test cases to test the `sumPartial` function written for Exercise 21. Write a test program to verify the correct operation of the `sumPartial` function for each test case.
23. Modify the `sumPartial` function written for Exercise 21 so that
    - If the function is invoked for an empty vector that an empty vector is returned as the result. Hint: MATLAB has an `isempty` function that returns 1 if the array is empty and 0 otherwise.
    - If the function is invoked for a start index less than zero or an end index greater than the length of the vector that an empty vector is returned as the result.
24. Determine an appropriate set of test cases to test the modified `sumPartial` function written for Exercise 23. Write a test program to verify the correct operation of the `sumPartial` function for each test case.
25. What will happen if the modified `sumPartial` function from Exercise 23 is invoked for a start index equal to the end index or a start index greater than the end index?

26. Write a MATLAB function named `readPositiveNumber` that will return a number greater than or equal to zero (a positive number or zero) read from the user. The function should ensure that the number returned is greater than or equal to zero and should keep reading in numbers from the user until a valid number is entered. Use the function definition below as a starting point.
```
function result = readPositiveNumber()
```
Note: it is appropriate to perform input in the `readPositiveNumber` function since that is the only operation that the function will perform. It will read in a positive number (or as many numbers as it takes to get a positive number) and return the positive number to the calling program.
27. Determine an appropriate set of test cases to test the `readPositiveNumber` function written for Exercise 26. Verify the correct operation of the `readPositiveNumber` function at the MATLAB Command line for each test case.
28. Write a MATLAB function named `buildPositiveVector` that will create and return a row vector of positive numbers. The function will take the number of elements and return the created row vector. Use the `readPositiveNumber` function developed for Exercise 26 to read in positive numbers; in other words invoke the `readPositiveNumber` function inside the `buildPositiveVector` function to read in each positive number. Use the function definition below as a starting point.
```
function result = buildPositiveVector(numberElements)
```
29. Determine an appropriate set of test cases to test the `buildPositiveVector` function written for Exercise 28. Verify the correct operation of the `buildPositiveVector` function at the MATLAB Command line for each test case.
30. Write a MATLAB function to compute the geometric mean of a set of numbers. You may assume the set of numbers is a row vector. The geometric mean of a set of numbers $x = (x_1, x_2, x_3, \cdots, x_n)$ is $G = (x_1 \cdot x_2 \cdot x_3 \cdot \cdots \cdot x_n)^{\frac{1}{n}}$; in other words the product of all the elements raised to the power of one over the number of elements. Solve this two ways: by iterating over the indices of the vector and using indexing, and scalar operations and by iterating over the data and use scalar operations.
31. Determine an appropriate set of test cases to test the geometric mean function from Exercise 30. Write a test program to verify the correct operation of the geometric mean functions for each test case.

Last modified Friday, September 26, 2014