EXAM 2 – WRITTEN PORTION

NAME _____

SECTION NUMBER _____

CAMPUS MAILBOX NUMBER _____

EMAIL ADDRESS _____@rose-hulman.edu

Written Portion	/ 48
Computer Portion	/ 52
Total	/ 100

USE MATLAB SYNTAX FOR ALL PROGRAMS AND COMMANDS YOU WRITE

Problem 1: (4 points) The following program (written_5.m) is supposed to perform element-by-element multiplication of the vectors `a` and `b` and then assign the result to `c`. However, when we run the code we see the error shown. Fix the error in the code.

```
clc
clear variables
a = [1 2 3 4];
b = [4 6 7 8];
c = a*b;
```

Command Window

```
Error using  *
Inner matrix dimensions must agree.

Error in written_5 (line 5)
c = a*b

fx >>
```

Problem 2: (4 points) Consider the main routine and function shown below. What is `d` after we run the code?

```
clc
clear variables
a = 1;
b = 2;
c = 3;
d = my_func_1(a,b,c);

function [output] = my_func_1(a,b,c);
output = [4*b; 2*c; 5*a];
```

Problem 3: (4 points) The code shown below is supposed to print integers starting at 1 (`a = 1`, `a = 2`, etc.) and their corresponding square root so long as the square root is less than 4. When we run the code, we find that

```
The square root of  1 is  1.00
```

is printed over and over until we kill the code. Mark the change(s) on the code so it will run properly.

```
clc
clear variables
a = 1;
sqrtofa = sqrt(a);
while sqrtofa < 4
      fprintf('The square root of %2.0f is %5.2f \n', a, sqrtofa);
      sqrtofa = sqrt(a);
end
```

Problem 4: (4 points) We have the main routine (written_2.m) and function shown below. When we run the main routine, we encounter the error shown. Fix the error in the code.

```
clc
clear variables
a = 2;
b = 4;
c = 6;

%  Compute d = 7*a + 5*b + 3*c:

d = my_func_2(a,b,c);

function [output] = my_func_2(a,b);
output = 7*a + 5*b + 3*c;
```

Command Window

```
Error using my_func_2
Too many input arguments.

Error in written_2 (line 9)
d = my_func_2(a,b,c);

fx >>
```

Problem 5: (4 points) What is `daniel` after running the code shown below?

```
clc
clear variables
daniel = [10:-2:3];
```

Problem 6: (4 points) Consider the code shown below. What is `vec` after running the code?

```
clc
clear variables
x = 1;
i = 1;
while x < 10
      vec(i) = 3*x;
      x = 2*x;
      i = i + 1;
end
```

Problem 7: (4 points) We have the main routine and function shown below. When we run the main routine, what will print?

```
clc
clear variables
a = 1;
b = 3;
c = 5;
d = my_func_3(a,b,c);
fprintf('d is %2.0f \n', d);

function [output] = my_func_3(b,c,a);
output = 2*a + 3*b + 4*c;
```

Problem 8: (4 points) Suppose we have defined the matrix `A` shown below. What is `lori` after running the code scrap shown below?

```
lori = A(2:3,1:2);
```

A

4x3 double

	1	2	3
1	1	2	3
2	4	5	6
3	7	8	9
4	10	11	12

Problem 9: (4 points) Consider the main routine (written_4.m) and function shown below. When we run the main routine, it is supposed to create a vector `vec` containing the squares of integers starting at 1 and ending at `n`. Instead, we get an error in the Command Window. Fix the error in the code.

Command Window

```
Attempted to access output(0); index must be a positive
integer or logical.

Error in my_func_4 (line 3)
    output(index) = i^2;

Error in written_4 (line 5)
vec = my_func_4(index,n);

fx >>
```

```
clc
clear variables
index = 0;
n = 4;
vec = my_func_4(index,n);

function [output] = my_func_4(index,n);
for i = 1:n
    output(index) = i^2;
    index = index + 1;
end
```

Problem 10: (4 points) The following code is supposed to generate the vector `x = [6 4 2]`. However, when we run the code, `x` is not created. Mark the change(s) on the code so it will run properly.

```
clc
clear variables
y = [3 2 1 0 -1 -2];
i = 1;
while y > 0
      x(i) = 2*y(i);
      i = i + 1;
end
```

Problem 11: (8 points) We have the main routine shown below:

```
clc
clear variables
angle = [0:720];
sine_wave = sind(angle);
[nrows,ncols] = size(sine_wave);
square_wave = make_square_wave(sine_wave, ncols);
```

In the main routine, we define a sine wave (`sine_wave`) and then call a function (`make_square_wave`) that transforms the sine wave into a square wave (`square_wave`) as follows: if a point in the sine wave has a value greater than or equal to 0, then a value of +1 is assigned for this point in the corresponding square wave; if the sine wave value is negative, then -1 is assigned for the square wave.

Complete the function below that may be used to convert the sine wave to a square wave. Do not change the function statement.

```
function [signal_out] = make_square_wave(signal_in, npoints);
```

EXAM 2 – COMPUTER PORTION

For this exam, you will create **TWO** m-files: a main program called `lastname_firstname_exam_2.m` (all lower case) and a function called `lastname_position.m` (all lower case).

The header section of your code should include your name, section number, and CM number; you do not need to do any other commenting. **There should be no output other than what is requested.**

Problem (52 pts)

Telemetry data for a car starting from rest and accelerating along a curved path was collected and stored in an Excel file `car.xls`. The car's position is measured using a radial coordinate r (in meters) and an angular coordinate θ (in radians). The car's speed v is determined from the rates of change of these coordinates and expressed in m/s. Time (measured in seconds) is stored in the first column of the Excel file, while r, θ, and v data are in the second, third, and fourth columns, respectively.

a) (14 points) Download the Excel file from the course website. Load the contents of the file into MATLAB. **Do not** hard code the dimensions of the loaded array. Plot the car's speed over time. Be sure to include good axis labels with units and a title.

b) (10 points) Write a function with 2 inputs and 2 outputs. The 2 inputs are the telemetry data r and θ, and the 2 outputs are the x-y positions of the car. The function will have as its first line the following:

```
function [x, y] = lastname_position(r, theta)
```

where `x`, `y`, `r`, and `theta` are x, y, r, and θ, respectively. The car's x-y positions are calculated according to the following equations:

$$x = r\cos\theta$$

$$y = r\sin\theta$$

The function inputs and outputs may be scalars or vectors, whichever you prefer.

c) (12 points) Use your function created in Part b to generate vectors of the car's x-y positions over time. Make a **new** figure (i.e., **do not** overwrite your first graph) to plot the path of the car (with x on the horizontal axis and y on the vertical axis). Set the axes so that x ranges from -30 m to 70 m, and y ranges from 0 m to 100 m. Again, make sure you include appropriate axis labels and a title.

(TURN OVER)

d) (16 points) Suppose we want to plot the car's path only for the time when it is traveling no more than 20 m/s. Have your program create **new** vectors of the car's x-y positions until just before the car's speed exceeds 20 m/s. **Do not generate any further values.** In a **new** figure, plot the corresponding path of the car, and use the same formatting as used in your plot in Part c.

When you are finished, put your main program (`lastname_firstname_exam_2.m`) and your function (`lastname_position.m`) in the Moodle assignment dropbox.

NOTE: All programming must stop 5 minutes before the end of the period. You will have 5 minutes after that to upload your files to Moodle if you need that time.