ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Department of Mechanical Engineering

ME 123 Comp Apps I

EXAM 2 – WRITTEN PORTION

NAME ____

SECTION NUMBER ________________________________

CAMPUS MAILBOX NUMBER _____________________

EMAIL ADDRESS _________________________________@rose-hulman.edu

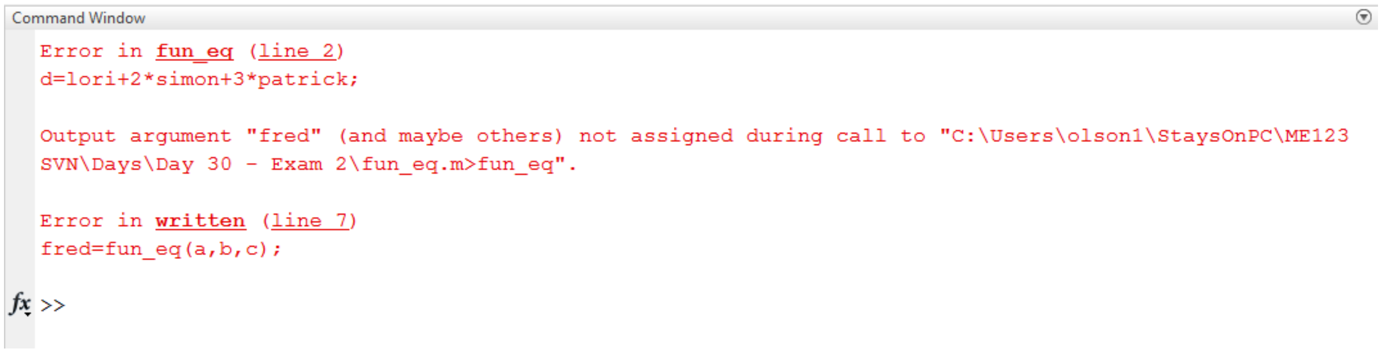
Written Portion	/ 40
Computer Portion	/ 60
Total	/ 100

USE MATLAB SYNTAX FOR ALL PROGRAMS AND COMMANDS YOU WRITE

Problem 1: (4 points) We have the main routine and function shown below. When we run the main routine we see the error shown. Fix the error in the code.

```
clc
clear variables
close all
a=1;
b=2;
c=3;
fred=fun_eq(a,b,c);

function [fred]=fun_eq(lori,simon,patrick);
d=lori+2*simon+3*patrick;
```


Command Window

```
Error in fun_eq (line 2)
d=lori+2*simon+3*patrick;

Output argument "fred" (and maybe others) not assigned during call to "C:\Users\olson1\StaysOnPC\ME123
SVN\Days\Day 30 - Exam 2\fun_eq.m>fun_eq".

Error in written (line 7)
fred=fun_eq(a,b,c);

>>
```

Problem 2: (4 points) The following code is supposed to keep multiplying `a` by itself and printing the value to the command window. It is supposed to run until `a` is bigger than 100. However, nothing prints to the screen at all. Mark the correction on the code.

```
clc
clear variables
a=2;
while a > 100
    fprintf('a=%3.0f \n',a);
    a=a*a;
end
```

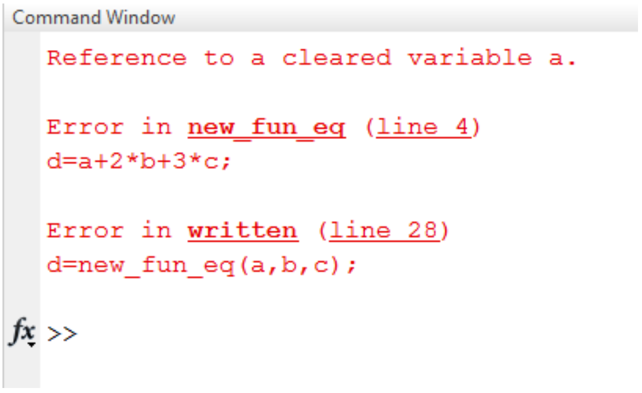
Problem 3: (4 points) Consider the code scrap shown below. We have loaded in the Sample_Data for project 2, and we wish to copy the first column of the data into a vector of x locations, the second column into a vector of y locations, and the third column into a vector of hole colors. Fill in the blanks in the code. (Do not change any of the existing code.)

```
clc
clear variables
Sample_Data=xlsread('Sample_Data.xlsx');
HoleX=Sample_Data(____________________);
HoleY=Sample_Data(____________________);
HoleColor=Sample_Data(___________________);
```

Problem 4: (4 points) We wrote the main routine and function shown below. When we run the main routine, though, we get an error in the command window. Mark the changes on the code so that it runs correctly.

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


function [d]=new_fun_eq(a,b,c);
clc
clear variables
d=a+2*b+3*c;
```



Command Window

```
Reference to a cleared variable a.

Error in new_fun_eq (line 4)
d=a+2*b+3*c;

Error in written (line 28)
d=new_fun_eq(a,b,c);

fx >>
```

Problem 5: (4 points) We are trying to print a table of the cubes of numbers, so long as the cubes are less than 100. No matter what value we start with (`n=2`, `n=1`, etc.) the code runs forever (until we kill it). Mark the changes on the code so it will run properly.

```
clc
clear variables
n=2;
ncubed=n*n*n;
fprintf('  n    ncubed \n');
while ncubed < 100
    fprintf('%3.0f    %3.0f \n',n,ncubed);
    ncubed=n*n*n;
end
```

Problem 6: (8 points) We have begun writing a function to find the distance between two points (x1,y1) and (x2,y2). Add the rest of the code to complete the function.

```
function [dist]=find_distance(x1,y1,x2,y2);
```

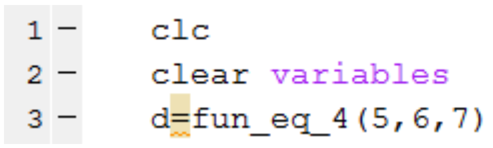
Problem 7: (4 points) Now let us suppose that we have done the previous problem correctly, what filename must we use for the function? (That is, what name must we use in saving the function?)

Problem 8: (4 points) Consider the main routine and function shown below. When we run the main routine, what will print?

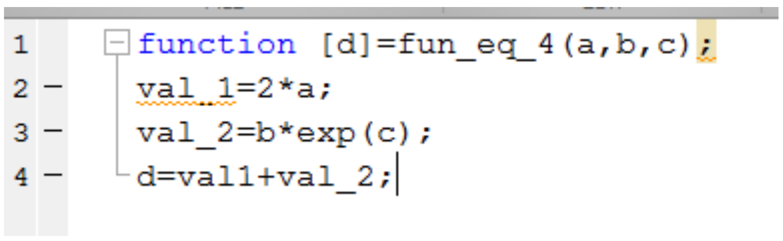
```
clc
clear variables
a=1;
b=2;
c=3;
d=third_fun_eq(5,6,7);
fprintf('d is %2.0f \n',d);

function [d]=third_fun_eq(a,b,c);
d=a+2*b+3*c;
```

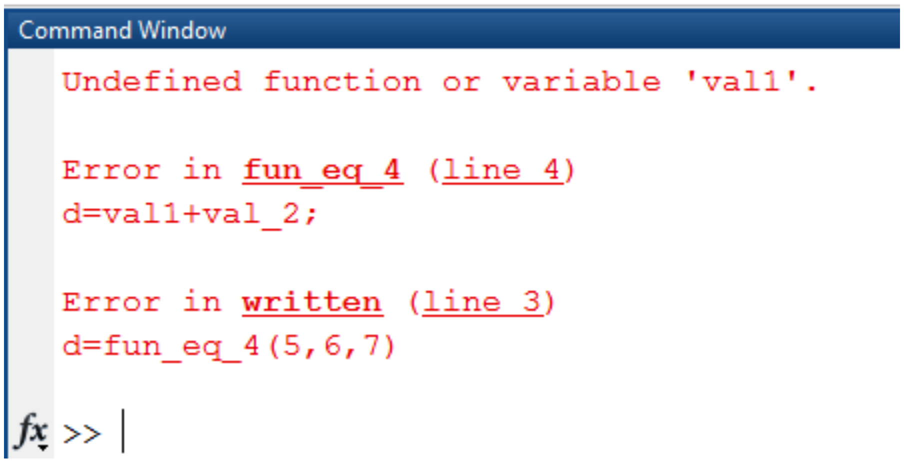
Problem 8: (4 points) Consider the main routine (written.m) and function shown below. When we run the main routine we get the following error. Fix the error in the code.



```
1 -    clc
2 -    clear variables
3 -    d=fun_eq_4(5,6,7)
```



```
1       function [d]=fun_eq_4(a,b,c);
2 -     val_1=2*a;
3 -     val_2=b*exp(c);
4 -     d=val1+val_2;
```



Command Window

```
Undefined function or variable 'val1'.

Error in fun_eq_4 (line 4)
d=val1+val_2;

Error in written (line 3)
d=fun_eq_4(5,6,7)

fx >>
```

Problem 9: (4 points) The following code scrap creates a vector named `vec`. Write out what the value of `vec` is after the code executes.

```
vec=[1:0.5:3];
```

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Department of Mechanical Engineering

ME 123 Comp Apps I

EXAM 2 – Computer PORTION

Name______________________________________

For this exam, you will create TWO m-files. A main program called `lastname_firstname_exam2.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.

When you are done, put both of your m-files in the Moodle dropbox.

Problem (50 pts)

A simple robotic device positions a tool at location (x, y) based on two angles (θ_1, θ_2).

The equations below show how to calculate the desired position (x, y) based on the angles and some other known parameters.

$$x = X_0 + b\,cos(\theta_1) + c\,cos(\theta_1 + \theta_2)$$

$$y = Y_0 + b\,sin(\theta_1) + c\,sin(\theta_1 + \theta_2)$$

where $X_0 = 180\ mm, Y_0 = 200\ mm, b = 200\ mm,$ and $c = 125\ mm$. It is permitted to "hard-code" these numbers, but no others (if you are not sure whether you are hard-coding, please ask your instructor).

1. **(9 points)** A set of angles θ_1 and θ_2 are stored in an Excel spreadsheet `Motion.xlsx`. The values for angle θ_1 are stored in the first column and the values for angle θ_2 are stored in the second column. THESE ANGLES ARE IN DEGREES. Load this Excel data sheet into your program; DO NOT plot the angles.

2. **(10 points)** Write a function with two inputs, the two angles θ_1 and θ_2, and two outputs, the values x and y as determined by the equations given above. This function will have as its first line:

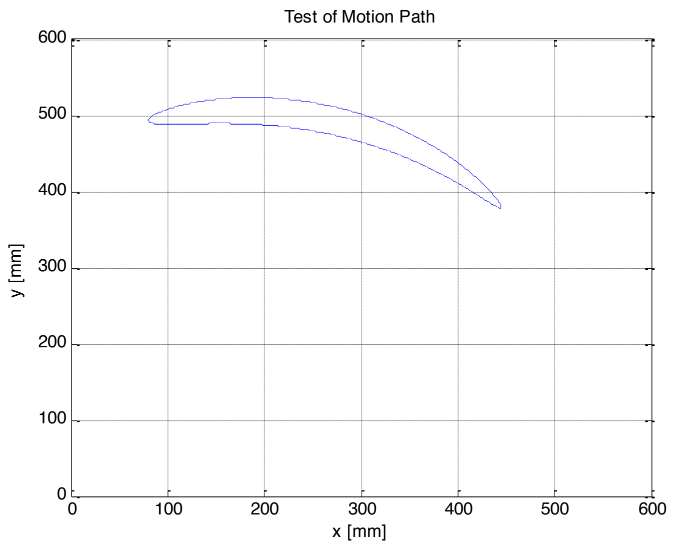
```
function [xpos,ypos] = lastname_position(theta1,theta2)
```

where `xpos` is x, `ypos` is y, `theta1` is θ_1, and `theta2` is θ_2. The inputs and outputs may be scalars or vectors, whichever you prefer.

NOTE: use your last name in place of the characters `lastname`.

3. **(10 points)** Use your function described in the previous part to generate a set of *x*- and *y*- coordinates corresponding to each row of the angle data described in Part 1. Use the *x*-*y* coordinates to plot the path of the robot; the resulting plot should look like the following (*i.e.* the same *x* and *y* axis ranges, labels, and gridlines).

   To learn how to turn on the grid lines, type `help grid`



4. **(15 points)** Suppose now that two barriers are put in place to limit the robot motion. The first barrier is placed at *x* = 400 mm (*i.e. x* cannot be greater than 400 mm); the second barrier is placed at *y* = 500 mm (*i.e. y* cannot be greater than 500 mm).

   Using the same set of angles you loaded in Part 1, your task is to have your program determine at what *x*-*y* location the robot first encounters a barrier.

   Starting from the first set of angles, have your program calculate *x* and *y* values until a barrier is encountered. Do not generate any further values.

   Once your program has stopped generating *x* – *y* coordinates, plot the motion path of the robot up until it reaches the barrier. Use a new figure to generate this plot, and use the same formatting as the plot in Part 3 (*i.e.* the same *x* and *y* axis ranges, labels, and gridlines).

   Hint: if you are using the same variable names for the *x* and *y* data in Parts 3 and 4, you'll need to clear the contents of the variables before starting Part 4. Another option would be to use a different set of variable names in this part.

5. **(6 points)** Have your program print the last *x* and *y* position generated in Part 4 to the command window. Use the following template:

```
The last point had x = xxx.xx mm and y = xxx.xx mm.
```