Duke University
Edmund T. Pratt, Jr. School of Engineering

EGR 53L Fall 2005

Test I

Rebecca A. Simmons
Michael R. Gustafson II

Name (please print) Answer Key

In keeping with the Community Standard, I have neither provided nor received any assistance on this test. I understand if it is later determined that I gave or received assistance, I will be brought before the Undergraduate Judicial Board and, if found responsible for academic dishonesty or academic contempt, fail the class. I also understand that I am not allowed to speak to anyone except the instructor about any aspect of this test until the instructor announces it is allowed. I understand if it is later determined that I did speak to another person about the test before the instructor said it was allowed, I will be brought before the Undergraduate Judicial Board and, if found responsible for academic dishonesty or academic contempt, fail the class.

Signature:

Problem I: [10 pts.] Basic Programming

Given some equation $ax^2 + bx + c = 0$, you can find two values for x,

$$x_{1\ 2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Write a one output, three input function you could use to take values for `a`, `b`, and `c`, then return a 1x2 matrix called `x` that contains the proper solutions. Your function should check if a is 0; if it is, your program needs to produce an error that says `a cannot be zero`. Beyond that, you can assume that `b` and `c` can be anything since Matlab understands complex numbers. Also, you can assume that the user is smart enough to enter three single-element (i.e. 1x1) variables. Finally, it does not matter in what order you store the solutions.

```
function x = input(a, b, c)

if a==0
     error('a cannot be zero');
end

x(1) = (-b+sqrt(b^2-4*a*c)) / 2 / a;
x(2) = (-b-sqrt(b^2-4*a*c)) / 2 / a;
```

```
% or as a script:
a = input('a: ');
b = input('b: ');
c = input('c: ');
if a==0
     error('a cannot be zero');
end

x(1) = (-b+sqrt(b^2-4*a*c)) / 2 / a;
x(2) = (-b-sqrt(b^2-4*a*c)) / 2 / a;
```

Name (please print):
Community Standard (print ACPUB ID):

Problem II: [20 pts.] Basic Functions

The area of an ellipse with semi-major axes `a` and `b` is π`*a*b` while the area of a circle is π`*r`^{2}. Write a *function* called `Around.m` that will accept up to two inputs. For zero inputs, the function should throw an error that there are no inputs. For one input, the program should print "Circle" to the screen and return the area of a circle with a radius given by the first input variable. For two inputs, the function should print "Ellipse" to the screen and return the area of an ellipse with the semi-major axes given by the two input variables. A sample run is included below:

```
>> Area1 = Around()
??? Error using ==> Around
No Inputs!

>> Area2 = Around(1/sqrt(pi))
Circle

Area2 =
    1.0000

>> Area3 = Around(3, 5/pi)
Ellipse

Area3 =
    15
```

```
function Area = Around(a, b)

if nargin==0
     error('No Inputs!')
elseif nargin==1
     r = a;
     Area = pi*a^2;
     fprintf('Circle\n')
else
     Area = pi*a*b;
     fprintf('Ellipse\n')
end
```

Notes:
1) use **nargin** *without any arguments* to determine number of input arguments. **nargin('Around')** will tell you how many inputs **Around** can take at most, which for this problem will always return 2

2) You cannot use **a** or **b** before checking to make sure they even exist - if nargin<2, **a** and **b** are not a part of the function's workspace and if you try to ask a question with them, you will get an error

3) Watch out for formatting issues such as spaces in elseif and writing π for pi

4) The command for an error is **error** and the argument is what Matlab tells you is the error

5) The name of the output that Matlab will return to the workspace is whatever comes after the word **function** so be sure to use that in your function

6) The **disp** command does not know formatting codes such as **\n**

7) An **if...elseif...else** tree needs one and only one end for the tree

8) The **else** part takes no logic

Name (please print):
Community Standard (print ACPUB ID):

Problem III: [20 pts.] Matrix Creation and Manipulation

For each of the following sections, either write the Matlab command required or answer the questions. Note that for the coding, efficiency will matter.

(a) Use the linspace function to create a vector identical to that obtained with the following statement:

```
x = -10:10
```

start, finish, # of points

```
x = linspace(-10,10,21)
```

(b) Use colon notation to create a column vector `y` with elements:

$$y = \begin{bmatrix} 9 \\ 6 \\ 3 \\ 0 \end{bmatrix}$$

```
y = (9:-3:0)'   NOT  9:-3:0', which uses 0'
```

(c) Create the following matrix `z` using the vector `y`:

$$z = \begin{bmatrix} 9 & 6 & 3 & 0 \\ 9 & 6 & 3 & 0 \\ 9 & 6 & 3 & 0 \end{bmatrix}$$

```
z = [y';y';y']  OR  [y y y]'
```

(d) Create a 4x3 matrix `LottaTwos` with every element equal to 2.

```
LottaTwos = 2*ones(4,3)  OR  1+ones(4,3)
```

Hard coding 12 2's is inefficient

(e) Change the element in column 2 and row 3 of `LottaTwos` to be equal to 4.

```
LottaTwos(3,2)=4
```

(f) Create the matrix `NoisyTwos` by adding an additional row to `LottaTwos` filled with random numbers between -2 and 2. Do this in one line.

```
NoisyTwos =
[LottaTwos ; -2+4*rand(1,3)]
```

(g) Create the matrix `SliceOfNoise` by extracting the 3rd through 5th row and every odd column column of `NoisyTwos`.

```
SliceOfNoise =
NoisyTwos(3:5,1:2:end)  OR  NoisyTwos([3,4,5],[1,3])
```

Name (please print):
Community Standard (print ACPUB ID):

Problem IV: [20 pts.] More Programming

Given the following Matlab commands:

```
x = 0:.5:3;
y = exp(x);
```

and the fact that $e^3 \approx 20.08554$,

(a) Write the code you would use to save x and y into a Matlab-format data file called `ColdPlay.mat`.

save ColdPlay x y / save ColdPlay x y -mat

(b) Write the code you would use to remove all variables from the workspace.

clear

(c) Write the command you would use to retrieve only `x` from the saved file.

load ColdPlay x / load ColdPlay.mat x -mat

(d) Write the command you would use to retrieve all variables from the saved file.

load ColdPlay / load ColdPlay.mat -mat

(e) Now that `x` and `y` are in the workspace, write the commands you would use to produce a properly formatted table on the screen, where `x` is in the first column and `y` is in the second column. You need to make sure there is a proper column heading. The `x` values should have one decimal point while the `y` values should have three. Your code should produce the table below, where the frame is given to show you the proper spacing. Indicate any spaces in your code with the ^ symbol. Note that part of the code is already given to you

			x							y
	0	.	0			1	.	0	0	0
	0	.	5			1	.	6	4	9
	1	.	0			2	.	7	1	8
	1	.	5			4	.	4	8	2
	2	.	0			7	.	3	8	9
	2	.	5		1	2	.	1	8	2
	3	.	0		2	0	.	0	8	6

```
x = 0:.5:3;
y = exp(x);
fprintf('^^^x^^^^^^y\n');
for k=1:7
     %your code here




end;
```

fprintf('%4.1f%8.3f\n', x(k), y(k)); OR
fprintf('%4.1f^%7.3f\n', x(k), y(k))

Name (please print):
Community Standard (print ACPUB ID):

Problem V: [20 pts.] Matlab Coding

Examine each group of Matlab statements. Are they correct or not? If they are correct, what do they output (represent a space with a '_ ')? If they are incorrect what is wrong with them? Make the corrections either on the code itself or rewrite the code to the right, and then show what the corrected code outputs. There may be multiple problems with each item. Also, there may be multiple possible corrections for a given piece of bad code - you only need to show one possible set of corrections.

(a)
```
temp = 110;
if temp > 100
        disp('Your temperature is high.');
if temp < 96
        disp('Your temperature is low.');
else
        disp('Your temperature is normal.');
```
elseif

end

Your_temperature_is_high.

(b)
```
a = [1, 1; 1, 1];
b = [1, 2; 3, 4];
c = a*b
```
OK $\begin{bmatrix}1&1\\1&1\end{bmatrix}*\begin{bmatrix}1&2\\3&4\end{bmatrix}=\begin{bmatrix}4&6\\4&6\end{bmatrix}$

$\begin{bmatrix}4&6\\4&6\end{bmatrix}$

(c)
```
u = 0:3;
v = (3:-1:0)';
w = u.*v
```
*u=(0:3)' OR v=(3:-1:0) OR w=u*v OR *w=u'.*v OR w=u.*v' OR w=u'*v'

: $\begin{bmatrix}0\\2\\2\\0\end{bmatrix}$.: [0 2 2 0] .: 4 .': $\begin{bmatrix}0&0&0&0\\3&2&1&0\\6&4&2&0\\9&6&3&0\end{bmatrix}$

(d)
```
A = eye(2)+1
```
OK

$\begin{bmatrix}2&1\\1&2\end{bmatrix}$

(e)
```
s = [1 2 2 4 5];
t = [6 2 3 4 5];
q = (s+t).*(s = t)
```
(s==t)

(s+t) → [7 4 5 8 10]

(s==t) → [0 1 0 1 1]

[0 4 0 8 10]

Name (please print):
Community Standard (print ACPUB ID):

Problem VI: [10 pts.] UNIX and LaTeX Processing

Assuming you have just logged in and opened a terminal window, give the proper UNIX commands needed to:

(a) Change into your EGR53 directory

```
cd EGR53
```

(b) Create and then change into a `testdir` directory

```
mkdir testdir
cd testdir
```

(c) Copy the skeleton file `TestSkel.tex` from Edwinna G. Rousseau's directory, `~egr53/skeletons/`, into your current directory

```
cp ~egr53/skeletons/TestSkel.tex .
```

(d) Rename `TestSkel.tex` so it is called `sample.tex`

```
mv TestSkel.tex sample.tex
```

(e) Assuming you have renamed the file properly, process `sample.tex` to produce a `.dvi` file

```
latex sample.tex / latex sample
```

(f) Preview the `.dvi` file

```
xdvi sample.dvi / xdvi sample
```

(g) Create a PostScript file named `output.ps` from the `.dvi` file

```
dvips sample.dvi -o output.ps
```

(h) Preview the `.ps` file

```
ggv output.ps
```