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

EGR 53L Fall 2008

Test I

Rebecca A. Simmons & Michael R. Gustafson II

Name (please print)________________________________

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

- Write *all* the MATLAB statements required to plot $\sin(x)$ versus $\cos(2x)$ from 0 to 2π in steps of $\pi/10$. The points should be connected by a black dotted line and each point should be marked with a black square marker. You do not need to label or title your plot, nor do you need to save it.

```
x = 0:pi/10:2*pi;
plot(cos(2*x), sin(x), 'k:s');
```

- Write an anonymous function to solve for the distance d between two points (x_1, y_1) and (x_2, y_2) on a Cartesian coordinate plane. Note that the distance can be given by:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Then, show the command used to have your function calculate the distance between a point (0.2, 0.4) and a point (0.1, 0.8).

```
% Have d as a four-input anonymous function
d = @(x1, y1, x2, y2) sqrt( (x2-x1).^2 + (y2-y1).^2 )
Distance = d(0.2, 0.4, 0.1, 0.8)

% OR - have each point entered as a 1x2 matrix
d = @(p1, p2) sqrt( (p2(1)-p1(1)).^2 + (p2(2)-p1(2)).^2 )
Distance = d([0.2, 0.4], [0.1, 0.8])
```

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

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

For each of the following sections, show what the matrices A, B, and C will look like at the end of the snippet of code.

(a)

```
>>clear
>>A(3, 3) = 7
>>B = A([3 2 1], [3 3])
>>C = B'
```

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 7 \end{bmatrix} \qquad B = \begin{bmatrix} 7 & 7 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 7 & 0 & 0 \\ 7 & 0 & 0 \end{bmatrix}$$

(b)

```
>>A=[-1 2; 20 5; 3 6]'
>>B=eye(2,3)
>>C=A.*B
```

$$A = \begin{bmatrix} -1 & 20 & 3 \\ 2 & 5 & 6 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 5 & 0 \end{bmatrix}$$

(c)

```
>>A=[1.6, -2.8; -4.2, 0.8]
>>B=fix(A)
>>C=floor(A)
```

$$A = \begin{bmatrix} 1.6 & -2.8 \\ -4.2 & 0.8 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & -2 \\ -4 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & -3 \\ -5 & 0 \end{bmatrix}$$

(d)

```
>>A = [zeros(1, 3); ones(1, 3); zeros(3, 1)']
>>B = A(2:5)
>>C = A(1:2, 2:end)
```

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \qquad C = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$$

(e)

```
>>A=linspace(-3, 3, 7)
>>B=logspace(-3, 3, 7)
>>C=length(A) * size(B)
```

$$A = \begin{bmatrix} -3 & -2 & -1 & 0 & 1 & 2 & 3 \end{bmatrix} \qquad B = \begin{bmatrix} 0.001 & 0.01 & 0.1 & 1 & 10 & 100 & 1000 \end{bmatrix} \qquad C = \begin{bmatrix} 7 & 49 \end{bmatrix}$$

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

Problem III: [20 pts.] Managing the Nitrogen Cycle

One of the Grand Challenges for Engineering involves managing the nitrogen cycle. To do this, you must first understand how the nitrogen cycle works. A paper on modeling[1] gives the following equation for the mineralization rate f as a function of the water content θ (Greek letter theta):

$$f(\theta) = \begin{cases} 0 & \theta \le \theta_W \\ \left(\frac{\theta-\theta_W}{\theta_L-\theta_W}\right)^m & \theta_W \le \theta \le \theta_L \\ 1 & \theta_L \le \theta \le \theta_H \\ f_s + (1-f_s)\left(\frac{\theta_S-\theta}{\theta_S-\theta_H}\right)^m & \theta_H \le \theta \le \theta_S \end{cases}$$

and states that "m is an empirical constant, θ_S is the saturated water content, θ_W is water content below which no activity is possible, and f_S is rate at soil saturation." θ_L and θ_H are constants describing the lower and upper boundaries of some optimal water content range. Basically, this means that m, θ_S, θ_W, θ_L, θ_H, and f_S are all constants.

Your first job is to write a function file that uses logical masks to take seven inputs (θ, m, θ_S, θ_W, θ_L, θ_H, and f_S) and return the appropriate mineralization rate. Only θ will be an array - all other values will be 1x1 constant matrices.

Your second job will be to write a script that plots f as a function of θ assuming:

$$m = 0.175 \qquad \theta_S = 0.494 \qquad \theta_W = 0.15 \qquad \theta_L = 0.31 \qquad \theta_H = 0.39 \qquad f_S = 1.5$$

Your θ values should go from 0 to θ_S and there should be 100 points to make the graph. You do not need to add a title or labels to this plot. Use a black solid line for this graph.

Function:

```
function f_out = NCycle(t, m, tS, tW, tL, tH, fS)

f_out = ...
    (t<=tW)         .* (0) + ...
    (tW<t & t<=tL) .* (((t-tW)./(tL-tW)).^m) + ...
    (tL<t & t<=tH) .* (1) + ...
    (tH<t & t<=tS) .* (fS + (1-fS).*((tS-t)./(tS-tH)).^m);
```

Script (first line already done for you):

```
m=0.175; thetaS=0.494; thetaW=0.15; thetaL=0.31; thetaH=0.39; fS=1.5;
theta = linspace(0, thetaS, 100);
f = NCycle(theta, m, thetaS, thetaW, thetaL, thetaH, fS);
plot(theta, f, 'k-')
```

[1] "Modelling water flow, nitrogen transport and root uptake including physical non-equilibrium and optimization of the root water potential." F. Lafolie. **Fertilizer Research**, Kluwer Academic Publishers. Vol 27, pp. 215-231. 1991.

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

Problem IV: [25 pts.] Advancing Health Informatics

One of the Grand Challenges for Engineering is to advance health informatics. In this problem, you will be doing that by helping to write an analysis and filing system for health data. A data set containing a rectangular array of blood pressure values has been saved, as a text file, to `BP.data`, and you have copied this file to your current working directory. You will be writing a script to load the data, process it, then potentially save the processed data into a MATLAB file. The original data file can be any size between 1x1 to 10x12 (and any rectangular size in between).

Start your script by loading the data. First, calculate the average of all the data in the matrix and store it in a matrix called `AvgPressure`. Next, calculate the highest value for all the data and store it in a matrix called `HighBP`. Once you have generated those two entries, you will report them back to the user in the following formatted way:

```
Average Pressure: NNN.NN
Highest Pressure: NNN.NN
```

where the N's represent locations where the pressures will print out. For example, if the average is calculated as 97.2343 and the high is 182.68234, the program should produce:

```
Average Pressure:  97.23
Highest Pressure: 182.68
```

Your program *must* properly line up the numbers - note that all pressures will be under 1000.00.

Finally, you need to ask the user whether the data should be saved - the user should be told that the options are `yes` or `no` and your program should make sure the user enters one of those two values - *anything* else and the program should keep asking until a valid entry is received. Should the user say `yes`, then the program should save the `AvgPressure` and the `HighBP` *only* into a MATLAB-formatted datafile called `Processed.mat`; otherwise, the program should just end. Note - in your code, be sure to *clearly* indicate any spaces in any strings you use using the underscore, or _ character.

```
load BP.data

AvgPressure = mean(BP(:));    % or mean(mean(BP))
HighBP      = max(BP(:));     % or max(max(BP))

fprintf('Average_Pressure:_%6.2f\n', AvgPressure);
fprintf('Highest_Pressure:_%6.2f\n', HighBP);

ToSave = input('Save (yes/no)? ', 's');
while ~(strcmp(ToSave, 'yes') | strcmp(ToSave, 'no'))
% OR: (~strcmp(ToSave, 'yes') & ~strcmp(ToSave, 'no'))
    ToSave = input('Save? Answer yes or no, please: ', 's');
end

if strcmp(ToSave, 'yes')
    save Processed AvgPressure HighBP
end
```

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

Problem V: [25 pts.] Engineering the Tools of Scientific Discovery

Part of the job of engineering the tools of scientific discovery will be to teach computers how to compute new functions by using old ones. For example, the value of $\sqrt{1+x}$ can be evaluated with the infinite series:

$$y = \sqrt{1+x} \approx \sum_{n=0}^{N-1} \frac{(-1)^n (2n)!}{(1-2n)(n!)^2 4^n} x^n$$

for $|x| < 1$ where N is the number of terms to use in the summation. Note that $a!$ is the factorial of a, and that MATLAB has a function called `factorial` that will compute this. For instance, `factorial(4)` will compute 4! or 4*3*2*1=24.

You are to write a script that asks the user for a number with a magnitude less than one and that checks to make sure the number does, in fact, have a magnitude less than one - if not, keep asking until the user gives valid input. Then ask the user how many terms to display - for this one, you may assume the user enters an integer greater than 0.

Finally, once you have a valid value for x and an integer N for the number of terms, have your program print out the first through Nth approximation to the square root. You should format the printout so that four digits are shown after the decimal point and so the computer displays how many terms have been used. For example, if the user gave 4 for N, the computer would calculate:

$$\begin{aligned}
N &= 1 & \sqrt{1+x} &\approx 1 \\
N &= 2 & \sqrt{1+x} &\approx 1 + \frac{1}{2}x \\
N &= 3 & \sqrt{1+x} &\approx 1 + \frac{1}{2}x - \frac{1}{8}x^2 \\
N &= 4 & \sqrt{1+x} &\approx 1 + \frac{1}{2}x - \frac{1}{8}x^2 + \frac{1}{16}x^3
\end{aligned}$$

and, if the user had put in 0.5 for x, the computer would display[2]

```
y = 1.0000 with 1 term
y = 1.2500 with 2 terms
y = 1.2188 with 3 terms
y = 1.2266 with 4 terms
```

```
x = input('Enter a number |x|<1: ');
while abs(x)>=1
    x = input('Magnitude less than 1, please! ');
end

N = input('Enter a positive integer: ');

y=0;
for n=0:(N-1)
    y = y + ( (-1).^n * factorial(2*n) ) / ...
        ( (1-2*n) * factorial(n).^2 * 4.^n ) * x^n;
    fprintf('y = %0.4f with %0.0f term', y, n+1)
    if n>0
        fprintf('s')
    end
    fprintf('\n')
end
```

[2]Small warning - there is a tiny bit of trickiness to this...