W1005 - Fall 2014
Homework 3

- Due by Friday 4pm (Oct. 17th).
- See submission instructions.
- Always include your name and UNI at the top of your submitted files.

1. The *greatest common divisor (gcd)* of two integers is the largest integer that divides them both. For example, the gcd of 20 and 5 is 5; the gcd of 20 and 15 is also 5.

   Implement a *recursive* function '**my_gcd**' which takes two input integers (m,n) and returns the gcd (g) as output .

   - Write your code for the case m >= n.
   - If m is less than n, you should warn the user (*warning* command) and then swap their values before proceeding.
   - You should check that m,n are both positive integers
   - Hint: writing down the stopping case and the recursive call can simplify implementation.

2. The problem consists of multiple parts. You should include the solutions to these parts in a single script called '**eq_solver.m**'. Make sure to precede each part with a comment.

   Given a system of 4 equations with variables x1,x2,x3,x4:
   $7 * x1 + 4 * x2 + 2 * x3 + 7 * x4 = 7$
   $3 * x1 + 4 * x2 + 5 * x3 + 8 * x4 = 2$
   $10 * x1 + 8 * x2 + 4 * x3 + 3 * x4 = 1$
   $8 * x2 + 6 * x3 + 7 * x4 = 5$

   Solve this system in two different ways.
   Using a single matrix multiplication, confirm that the values for x1,x2,x3,x4 yield the proper solutions to the system.
   Note: you can hard-code the system in your script

3. Recall the lecture slides on random numbers. Below we use some of the commands to generate random data.

   a) Write a function ‘**rgauss**’ which takes as input two uniformly distributed numbers (u1,u2) and outputs two standard normally distributed random numbers (z1,z2). You should use the equations in the relevant lecture slide.

      To make sure that part (b) below is easier to solve, ‘rgauss’ should accept either scalars or vectors/matrices of numbers (u1,u2) and correspondingly output scalars/vectors/matrices (z1,z2).

   b) Write a function ‘**rdata**’ which takes two input arguments (N,K) and generates random K-dimensional data as follows:
      - The function generates a set of N data points (each point has K attributes/dimensions, point = row of returned matrix)
      - The function makes calls to the ‘rgauss’ function to generate each point.
      - For simplicity assume that K is an even integer
      - You need to generate uniformly distr. numbers before calling ‘rgauss’
      - The function returns a matrix D with the N random data points that were generated.

   c) Write a function ‘**data_sample**’ which has at least one input argument (S), and performs the same operation on each subsequent argument (if they exist):

      - You can assume that each subsequent (potential) input argument is a matrix of data points (see part b).
      - For each data matrix, the function selects a random sample of size ‘S’ and returns a matrix of only the sampled data points.
      - Store your sampled data matrices in a cell array (see hint)
      - Use ‘*randperm*’ to choose your random sample. Choose any settings to your generator (rng), but do NOT use the default ones.

- You can assume that the number of points in each data matrix is at least 'S' (i.e. there are no errors in the argument). But the exact number can vary from matrix to matrix.
- Hint: use *varargin, varargout*

4. Recall the game from HW2:
   Choose any positive integer (1,2,…), call it X.
   If X is an even integer you divide X by 2.
   If X is an odd integer you multiply X by 3 and add 1.
   You continue this procedure until at some point you get 1 as your next integer. At that point you stop.

   a) Write a function named **'game3'** which has one input argument {N}, and one output argument {A}. Your function should do the following:

      You are going to play the game in reverse order. So starting from 1 you will backtrack:

1	2	4	8	16	32	64	128	256	512
									85
							21	42	84
					5	10	20	40	80
									13
							3	6	12

      Note: we ignore the cycle {1,2,4}.
      The input parameter N is the number of steps you are going to backtrack. For each step, you should store all the integers you reach in the corresponding element of your cell array A.
      For example, element 5 of A should be 16, element 6 of A should be the vector [5, 32].

5. Write a function '**add_grades**' which takes as input a cell array (G), an integer (N, which denotes row number) followed by (name, hw1,hw2, mt) which denote name of each student (string), and their grades for hw1, hw2, and midterm (all numbers).

   Your function should return a single variable (G) which is the updated cell array with the information entered in the appropriate row (N).

   Your function should accommodate the case when the user calls add_grades with fewer input arguments than indicated (2 or more instead of 6). In this case, you should set the value of G in the appropriate row/column to []. For example, if the user only uses 5 input arguments, you should set the midterm grade to [].

Your zip folder should include the following files:

my_gcd.m
eq_solver.m
game3.m
rgauss.m
rdata.m
data_sample.m
add_grades.m