W1005 - Fall 2014
Extra Homework

- Due by Friday 4pm (Nov. 14th).
- See submission instructions.
- Always include your name and UNI at the top of your submitted files.

1. Recall the Middle Square Method from the math lecture notes. You will implement this random number generator as follows:

   a) Write a function named **'msrng.m'** which has 3 input arguments: V (the seed), n (number of digits), npts (length of random sequence), and an output argument R (cell array).
   b) Your function should implement the steps of the given pseudo-code (in the notes).
   c) For each iteration you should record the actual number generated in the first column, and the next padded number (string) in the second column of the array R. For example, the call 'msrng(540,4,2)' should generate:
   R{1,1} = 540, R{1,2} = '00291600'
   R{2,1} = 2916, R{2,2} = '08503056'

   Note: you can assume that n is even and that there are no errors in the input parameters.
   Hint: there are multiple ways to pad a number with zeros. One approach is described in the doc for the sprintf function.
   Hint: use str2num & num2str functions.

2.
   a) Write a function named '**read_my_line**' which has one input parameter ('S') and one output parameter ('ret_line'). Your function should do the following:
   b) Suppose 'S' is a struct with 4 fields: {file_dir, file_name, N, my_line}. Check that:
      1. 'S' is indeed a struct.
      2. The fields 'file_dir', 'file_name' are (non-empty) strings

c) You should use the '*error*' function to indicate an error if your checks above fail.
d) Now suppose you have a directory whose name is stored in 'S.file_dir' which contains a file whose name is stored in 'S.file_name'. You should assume that you have a simple text file which contains text (NOT numeric data).
e) Your function should open the file (hint: use *fopen*), and read 'S.N' number of lines (rows) from the file into a cell array 'C'. Each element of C should contain one line from your file. You can assume that there are no errors in the parameter 'S.N', that is, it cannot exceed the number of lines in your file.
f) Suppose 'S.my_line' specifies some line number. The output parameter 'ret_line' should be set to this line, where all the letters are lower-case (you should return a string).
g) Verify that your code works by saving some text into a file (created in notepad / textedit) and saving the file with extension '.txt'. You should attach the file (call it '**test.txt**') with your code.

3.

a) Write a function named '**data_plot**' which has 3 input parameters (a,b,c). Your function should do the following:
b) The values of (a,b) denote the range of your domain (x-values). Generate 200 linearly spaced points on the domain (hint: help linspace)
c) Compute the function value f(x) for each point on your domain. Repeat this for the following 4 functions:

{ 5*cos(x)^2, x^(0.5) + 1/x, x^2-2*x+4, 1/ln(x) }

Each set of values f(x) should be a <u>column</u> vector (named f1,f2,f3,f4).
d) Plot each function over the specified domain. You should generate four <u>distinct plots</u> on the <u>same figure</u> (do not plot one function over the other). Each function should be plotted in a different color (specified in the cell array 'c' and corresponding index, 1st function – 1st index) and do NOT use the default line-width.
e) Each distinct plot should have its own title with the corresponding function name (e.g. 'inverse natural logarithm' or '1 over ln x')
f) Once you've completed the plot, save your figure to the file 'data.fig' (hint: 'help saveas').

Your zip folder should include the following files:

- msrng.m
- read_my_line.m
- data_plot.m
- data.fig
- test.txt