GG250 Lab 6
Loops

This lab uses loops to sort 2D arrays.

In all the exercises:

a) **Make sure you clearly understand the problem.**
b) **Work the following problems "by hand"** (you can use Matlab as a calculator):
c) **Prepare a flow chart**
d) **Develop the code**. **Before each "active command" in the code (or before a small set of associated commands), insert comments saying what the line (or set of lines) does**.
e) **Test the function**
f) **Attach to e-mail to**
gg250-lab@hawaii.edu
the following items
* your Matlab functions
* your WORD file with the flowchart

Use the naming scheme established in the class for your functions (e.g., gg250_lab06_1a_martel.m).

Exercise 6_1a (function file)
Prepare a function that sorts the numbers in a table (a "square magic matrix") so that the numbers increase as you read the table like you read a paper (increasing from left to right, with the numbers increasing row-by-row proceeding from the top to the bottom. Use a doubly nested loop in the function. Build upon the swapping procedure in the following Matlab code available on my website:
gg250_lab_06_try.

Write the function in a way so that after each swap the revised matrix is printed out (this is so you and the instructors can follow the progress of the code). Do this by leaving the semicolon (;) off the end of certain lines in the code. Test the code for a 3x3 magic matrix, but allow a magic matrix of any size to be used. Include a line plotting the value of the elements against their position in the array.

The starting and ending matrices for the test should look like this:

8	1	6
3	5	7
4	9	2

$\Rightarrow$

1	2	3
4	5	6
7	8	9

Use no more than 20 active lines (mine has 15).

I found the following Matlab commands helpful: magic, (:), reshape, and (‘), for transposing a matrix. DO NOT USE THE MATLAB function called "sort"!

The head of the code should look like this:

```
function [A,D] = gg250_lab_06_1a_yourname(n)
% function [A,D] = gg250_lab_06_1a_yourname (n)
% Sorts the elements in a magic nxn matrix
% so that the numbers increase left-to-right
% and top to bottom.
% Input parameters
% n = number of rows in magic matrix
% Output parameters
% A = original magic matrix
% B = final ordered matrix
% Example
% Example
% [A,D] = gg250_lab_06_1a_yourname (3)
```

Exercise 6_1b (function file)

Write a new function that uses no loops that sorts the elements of a magic matrix as described above, using no more than 5 active lines (mine has 2). You do not need to print out the matrix as it is updated for the final product. If you do this correctly, this code will run much faster than the preceding one. I found the following Matlab commands helpful: magic, (:), sort, reshape, size, and (‘).

The head of the function should look something like this:

```
function [A,B] = gg250_lab_06_1b_yourname(n)
% function [A,B] = gg250_lab_06_1b_yourname(n)
% Sorts the elements in a magic nxn matrix
% so that the numbers increase left-to-right
% and top to bottom.
% Input parameters
% n = number of rows in magic matrix
% Output parameters
% A = original magic matrix
% B = final ordered matrix
% Example
% [A,B] = gg250_lab_06_1b_yourname(4)
```