Programming Assignments

Introduction to Programming with MATLAB

Lesson 4

- Unless otherwise indicated, you may assume that each function will be given the correct number of inputs and that those inputs have the correct dimensions. For example, if the input is stated to be three row vectors of four elements each, your function is not required to determine whether the input consists of three two-dimensional arrays, each with one row and four columns.
- Unless otherwise indicated, your function should not print anything to the Command Window, but your function will not be counted incorrect if it does.
- Note that you are not required to use the suggested names of input variables and output variables, but you must use the specified function names.
- Finally, read the instructions on the web page on how to test your functions with the auto-grader program provided, and what to submit to Coursera to get credit.

1. Write a function called **`quadrants`** that takes as its input argument a scalar integer named **`n`**. The function returns **`Q`**, a **`2n`**-by-**`2n`** matrix. **`Q`** consists of four **`n`**-by-**`n`** submatrices. The elements of the submatrix in the top left corner are all 1s, the elements of the submatrix at the top right are 2s, the elements in the bottom left are 3s, and the elements in the bottom right are 4s.

2. Write a function called **`checkerboard`** that takes as input two positive integer scalars, **`n`** and **`m`**, in that order. The function must create and return **`board`**, which is an **`n`**-by-**`m`** matrix. Every element of board is either 0 or 1. The first element, **`board(1,1)`** is 1. No direct neighbors in the matrix, vertically or horizontally, can be equal. That is, a 1 element cannot have 1 immediately preceding or following it in the same row or column.

3. Write a function called **`randomness`** that takes three input arguments: **`limit`**, **`n`**, and **`m`**, in that order. The function returns an **`n`**-by-**`m`** matrix of uniformly distributed random integers between 1 and **`limit`** inclusive. You are not allowed to use **`randi`**, but you can use **`rand`**. You will also find the built-in function **`floor`** useful for obtaining integers. To make sure that your result is indeed uniformly distributed, test the output of the function by using the built-in function **`hist`**, which plots a histogram.

4. Write a function called **`mtable`** that returns **`mt`** an **`n`**-by-**`m`** matrix corresponding to the multiplication table (**`n`** is the first and **`m`** is the second input argument). That is, the element of **`mt`** at row **`ii`** and column **`jj`** equals to **`ii*jj`**. The function also has a second output, **`s`**, a scalar that equals the sum of all the elements of **`mt`**.

5. Write a function called **`identity`** that creates a square identity matrix, which is a matrix whose elements are 0 except for the elements on the diagonal (from top left to bottom right) which have a value of 1. The diagonal consists of those elements whose row and column indexes are the same: (1,1), (2,2), etc. The function takes one positive integer input argument, which is the size of the matrix, and returns the matrix itself as an output argument. For example, **`identity(4)`** must return a 4-by-4 identity matrix. You are not allowed to use the built-in **`eye`** or **`diag`** functions. (Hint: you can index into a matrix with a single index and MATLAB will handle it as if it was a vector using column-major order.)