CS1112 Exercise 9 Name: ______________________ NetID: ________

You have until *Sunday, 10/23, at 9pm* to finish this exercise. Get problem 1 checked off by a consultant or TA, then submit Problems 2 and 3 on MATLAB *Grader*.

1 Subarrays (get checked off by TA or consultant)

Type the following expressions in the Matlab *Command Window*. Write the resulting array or answer the question on the blank.

```
m = rand(6,5)

a = m(:,2)      % What does the colon specify when used in place of an index? ____________


b = m(2:3,:)    % __________________________________________________________________________

p = rand(6,5,3) % This is a 3-dimensional array

[nr, nc, np]= size(p)  % ___________________________________________________________________

c = p(:,:,2)    % Is this a matrix (2-d) or a 3-d array? __________________________________

d = p(4,:,2)    % Is this a vector, matrix, or 3-d array? _________________________________
```

2 Cumulative sums (Answer in MATLAB Grader https://grader.mathworks.com)

Implement the following function as specified. Do *not* use any built-in functions other than `size`. If you are struggling with this problem, first write out an example by hand.

```
function A = matrixCSums(M)
% `M` is a numeric matrix and `A` has the same size as `M`.  Assume `M`
% is not empty.  Each element in `A` is the sum of the corresponding
% element in `M` and all the elements above it.  Example:
%   M = [ 1 3; ...              A = [ 1  3; ...
%         4 5; ...     then           5  8; ...
%        -7 2]                       -2 10]
% Do not use any built-in functions other than `size()`.
```

3 Two-dimensional interpolation

First use the full Matlab environment to develop, test, and debug your code; then submit your solutions in Matlab Grader.

When you enlarge an image, you are actually adding data points among the existing data (pixels). How do you get the additional data points? One way is to interpolate from the neighboring points—take the average value. First, consider a simple case of one-dimensional interpolation, we add a data point between neighboring pairs of existing data points by taking the simple average. For example,

`2.0 1.0 1.0 2.0` becomes `2.0 1.5 1.0 1.0 1.0 1.5 2.0`.

In 2-d interpolation, work with one dimension at a time. For example, given a matrix

```
2.0  1.0  1.0  2.0
6.0  5.0  4.0  3.0
5.0  5.0  5.0  4.0
```

First we can add a column between two neighboring columns, so the matrix becomes 3×7:

```
2.0  1.5  1.0  1.0  1.0  1.5  2.0
6.0  5.5  5.0  4.5  4.0  3.5  3.0
5.0  5.0  5.0  5.0  5.0  4.5  4.0
```

Then add a row between neighboring rows, so the final matrix will be 5×7:

```
2.0  1.5  1.0  1.0  1.0  1.5  2.0
4.0  3.5  3.0  2.8  2.5  2.5  2.5
6.0  5.5  5.0  4.5  4.0  3.5  3.0
5.5  5.2  5.0  4.8  4.5  4.0  3.5
5.0  5.0  5.0  5.0  5.0  4.5  4.0
```

Write two functions for doing 2-d interpolation as discussed above: (a) `interpolate2d_nv` uses non-vectorized code; (b) `interpolate2d_v` uses vectorized code (work with whole columns one at a time; then whole rows). Do not use built-in functions `mean`, `sum`, and `linspace`.

```
function newM = interpolate2d_nv(M)
% Perform 2-d interpolation on the real-valued data in nr-by-nc matrix M
%   where nr>1 and nc>1.
% The interpolated data are added between existing data points so newM is
%   (2*nr-1)-by-(2*nc-1).  Use the simple average as the interpolated  value.
% Do NOT use built-in functions mean, sum, and linspace.
% NON-VECTORIZED implementation.
```

The vectorized version has the name `interpolate2d_v` but the same specification as above other than using vectorized code. In addition to submitting your code on Matlab Grader, ask a member of the course staff to look over your vectorized solution to ensure that you are using vectorized code effectively.