CSCE155N Lab 02

1. Lab Objectives

a. Introduce arrays and array operators.
b. Differentiate array operators and matrix operators.
c. Create, access, modify, and resize arrays.
d. Use arrays and logical arrays as indices into another array.
e. Solve computational problems that involve array manipulation.
f. Formalize science/engineering problems as matrix or array computations.
g. Practice formatting output.
h. Introduce vectorized code.

2. Prior to Lab

a. Read the laboratory handout, and verify the examples in the practice section.
b. Read chapters 1, 2, 4, and 5.

3. Practice

a. Array Creation: Type each of the following Commands into the Command Window and observe the output.

```
array = [ 1 2 3 4 ]
a = [ 0 1 + 7 ]
b = a( 2 )
a( 2 ) = 12; disp( a );
b = [ a( 2 ) 7 a ]
x = [ 1 : 2 : 10 ]
y = 1 : 2 : 10
ylen = length( y ); disp( ylen );
g = 1 : 4; k = g; disp( k );
```

b. Matrix Creation: Type each of the following Commands into the Command Window and observe the output.

```
x = [ 1 2 3; 4 5 6 ]
y = [ 1 2 3; 4 5 ]
x( 2 )
x( 2 , 1 )
x( 2 , 3 ) = 20;
a = 2 + x
a = 2 - x
a = 2 * x
a = x / 2
```

```
b = zeros( 2 )
c = size( b )
d = zeros( 1 , 2 )
e = ones( 3 )
f = eye( 4 )
```

c. `fprintf()` and `sprint()`: Type each of the following Commands into the Command Window and observe the output.

```
printf( '%f\r\n' , pi );
fprintf( '%.2f\r\n' , pi );
fprintf( '%05.2f\r\n' , pi );
fprintf( '% 5.2f\r\n' , pi );
fprintf( '% 5d' , 10 );
out = sprintf( '%f' , pi ); disp( out );
out = sprintf( '%.2' , pi ); disp( out );
out = sprintf( '%05.2f' , pi ); disp( out );
out = sprintf( '% 5.2f' , pi ); disp( out );
out = sprintf( '% 5d' , 10 ); disp( out );
```

4. Activities

a. Download all files from URL NEEDED to your Z:csce155n\lab02 or equivalent directory.

b. Array Operations

   i. Modify AtimesB.m so that the results of A x B is correctly computed and displayed

   ii. Modify CtimesD.m so that the result of C x D is correctly computed and displayed

   iii. Modify CplusD.m so that the result of C + D is correctly computed and displayed. Note that not all of these operations are straight forward. Leave comments explaining what steps you used to accomplish the required task and why. Example values for A, B, C, and D are provided below, and should be used for testing purposes

$$A = \begin{bmatrix} 10 & 2 & 4 \\ 5 & 6 & 2 \\ 3 & -3 & 1 \end{bmatrix} \; B = \begin{bmatrix} 5 & 9 & -12 \end{bmatrix} \; C = \begin{bmatrix} 5 & 10 & 4 \\ -3 & 5 & -3 \end{bmatrix} \; D = \begin{bmatrix} 1 & 2 \\ 5 & 10 \\ 8 & 6 \end{bmatrix}$$

c. 3D Matrices

   i. Modify makeMatrix.m so that the function creates and displays a matrix of your choice with three dimensions. There are several ways to accomplish this, but the zeros() or ones() Commands may be helpful

d. Vector Manipulation

   i. Input the Command `time = clock` into the Command Window, and observe the result in your workspace. Note that the hours, minutes, and seconds are displayed in the fourth, fifth, and six indices of the array

ii. Modify timeToClock.m so that the function displays only the correct *whole* number of hours, minutes, and seconds. The built in function `fix()` can be used to get the integer part of a fractional number

e. Mathematical Operations over Vectors and Basic Plotting
   i. Modify plotCosine.m to create a vector of 50 evenly spaced points between $-\pi$ and π, calculate the cosine of those points, then plot those points versus their cosine
   ii. Label the plot's x and y axes
   iii. Include a legend for the curve
   iv. Save your plot as an EPS file, with the filename plotCosine.eps
   Hint: MATLAB has the built-in functions `linspace()`, `cos()`, `plot()`, `xlabel()`, `ylabel()`, and `legend()`

f. Plotting Several Curves
   i. The sales (in billions of dollars) for two separate divisions of XYZ Corporation for each of the four quarters of 2016 are stored in a file called salesFigures.mat
   ii. Modify plotSales.m, to load the file salesFigures.mat
   iii. Separate this 2 x 4 matrix into vectors for the two divisions
   iv. Create a single plot that contains two curves (one for each division)
   v. Each division's curve should be a different color
   vi. Label the plot's x axis and y axis
   vii. Include a legend for the curve
   viii. Save your plot, as an EPS file, with the filename plotSales.eps
   ix. Hint: MATLAB has built-in functions `load()` and `hold()`

g. Smallest
   i. Modify smallest.m such that it displays the smallest of three input variables using `if`, `elseif`, `else`, and `end` statements. Be aware of the use of `>`, `<`, `>=`, `<=`, `==`, `&&`, and `||` operators. Note that although `min()` and `max()` functions are built-in to MATLAB, you may not use them in this exercise.

h. Hamming Distance: Hamming distance is the distance between two strings of equal length, measured by the number of letters in the string that are not identical.
   i. Modify hammingDistance.m such that it correctly calculates the hamming distance of any two strings of equal length. The `sum()` Command and `==` operator will likely be useful. Note that a single equals sign is used to set one variable equal to another, while two are used to check for equality, and create a logical array.

i. Making Change
   i. Modify makingChange.m to convert the cents given as input into the minimum number of American coins needed to make change for that amount. The `fix()` or `floor()` Commands may be helpful.

j. Leap Year (Extra Credit)
    i. A given year is normally a leap year if it is a multiple of four. However, on years divisible by one hundred, this rule is ignored, except in years also divisible by four hundred. For example, 2020 will be a leap year, but 2100 will not be, but the year 2000 was.
    ii. Modify leapYear.m such that it correctly calculates whether or not any given year will be a leap year.
    iii. `If` statements and the `mod()` Command, which returns the remainder after division of two numbers will be helpful.