In general, Matlab is a very safe computing environment. Feel free to try things out to see how they work, and feel free to talk with your Prof. if you have any questions about the results!

**IMPORTANT NOTE:** You will need to demo your work (in class) with your Prof. to get it “checked off” as part of this lab assignment. Matlab’s `diary` command may be useful to you—please look it up to make sure you know how it works before you begin your work! (Note that it has the function of toggling on and off—you may want to test to make sure logging is on before you type commands that you want saved!)

1. **Review!** Look over the lecture notes from our last class meeting. Make sure you understand how the Matlab examples were generated, and make sure you know the answers to questions such as these:

   - What’s the purpose of a semi-colon at the end of a line of Matlab code? What could happen if you omit them?
   - What’s the difference between the regular multiplication operator `*` and the dotted multiplication operator `.*`? (Important CS346 point of style: Please treat non-dotted operations as the default—only use dotted operators when needed!)
   - What do the following Matlab functions do: `clear`, `clc`, `diary`?
   - What are three ways to find help with Matlab commands from within Matlab (without using Google or other web search tools)?

2. **Array Indexing!** Matlab streamlines many matrix operations, and its indexing scheme facilitates that. These are some little examples to illustrate Matlab’s array indexing—please try other examples on your own, too!

   (a) Using a statement of the form `a = [ ... ]`, create an array `a` with 10 elements such that five of them are positive integers and five are negative integers. Put any five positive and five negative integers inside the square brackets, separated by spaces.

   (b) Use the `begin:step:end` syntax presented in the lecture notes, create an array (vector, list—we’ll consider them all the same in Matlab right now) with the odd numbers between 1 and 9 inclusive. Call the array `i`; create it using a statement of the form `i = #:#:#`.

   (c) What does `a(i)` equal? Type `a(i)` in the command window to see.

   (d) Using these techniques, display an array that consists of the cubes of the even-indexed elements of `a` (i.e., it should be `[a(2)^3 a(4)^3 a(6)^3 a(8)^3 a(10)^3]`).

   (e) Also display an array that consists of the negatives of elements 3 through 7 in `a` (i.e., -1 times each of the elements, displayed as a 5-element array).
3. Logical operators!

(a) Look up Matlab documentation for the logical operators and, or, and not.

(b) For the examples below, we'll use randomly generated arrays of integers. Look up Matlab's `randi` command, and using `randi`, generate two 8-number arrays of randomly chosen numbers between -10 and 15 (inclusive). As part of the exercise, you may not use an IMIN argument to `randi` (please read the documentation for `randi` carefully to see what's being referred to—avoid using any form of `randi` with an IMIN parameter) for this—your calls to `randi` should only return values 1 or greater, but you should end up with arrays containing values from the proper range. Store the arrays in variables `x` and `y`.

(c) Now, see what the following commands do:
   i. \( x < y \)
   ii. \( x == y \)
   iii. \( x \mid y \)
   iv. \( x \& \neg y \)
   v. \( (x > -1) \& (y < 2) \)

(d) Matlab can access array elements using logical indexing—using the results of logical operators. Try the following commands, and be sure to understand what they do:
   i. \( x(x > 4) \)
   ii. \( y(x > 4) \)
   iii. \( x((x < 2) \mid (x > 5)) \)

(e) Now, for some exercises! Give a command that will:
   i. Display the array that is identical to `x` but with all negative values set to 0. (There are multiple ways to do this! Please try to find one that does not use additional variables—so, without copying `x` into some other variable—and does not change the value of `x`!)
   ii. Create an array `z` that contains exactly the values of `x` that are negative numbers.
   iii. Display the array that is identical to `y` except all even-valued elements are multiplied by 2. (You might look up the `rem` function for use in this!)

4. Positivity! How can you return the number of positive entries in a matrix?

Please think about elegant solutions for this—this can be done in a single line of code (with composition of functions), without a loop! (Your code should work for scalars, arrays, and 2D matrices. For this exercise, your answer doesn't need to work for matrices of dimension higher than 2.)
5. **Logistic Hyperbole!!!** These exercises work with the logistic function we discussed in class and the *hyperbolic tangent* function as examples for practicing graphing with Matlab’s `plot` command.

(a) Graph the hyperbolic tangent function \( \tanh \) (see the `tanh` help in Matlab’s documentation).

(b) In a different figure from the \( \tanh \) graph, graph the logistic function (we’ll call it \( s \)) that we discussed in class \( s(t) = 1/(1 + e^{-t}) \). To plot this in a different figure window, type the command `figure` to open a new figure window; then, the next `plot` command will display in that new window.

(c) What do you observe about the \( \tanh \) function and logistic function \( s(t) \)?

(d) Find the definition of the hyperbolic tangent function (in terms of constants and addition, substraction, multiplication, division, and exponentiation functions only—not in terms of other hyperbolic functions!) using Matlab’s documentation. How does this inform your observation in exercise 5c above?

(e) Now, graph both functions \( \tanh \) and \( s \) on the same axes. Be sure to plot them in different colors, so you can tell them apart; also, put a label on the \( y \) axis to say which color is for which function. (Read the help pages for `plot` to find out how to add colors to plots, and look at `xlabel` and `ylabel` for labeling axes.) To do this, first plot \( \tanh \), then type the command `hold`, then plot \( s(t) \)—the `hold` command will make the second plot go on the same axes as the first.

(f) The label in that graph probably shouldn’t be placed on the axis. Look at documentation for the `text` command, and put helpful text on the figure that says which color is associated with which function.

Then, look at documentation for the `legend` command, and use it to add a legend to your figure that shows which color is associated with which function.