The purpose of this assignment is to gain familiarity with basic arrays and matrices. You can learn about these with the Introducing Julia page.

To practice using basic arrays, you'll create the full NACA 4-series airfoil coordinate function, which will call the functions you wrote in activities 1 and 2.

NACA 4-series Airfoil Coordinates *The NACA 4-series airfoil coordinates are obtained by adding and subtracting half the thinkness from the mean camber line. That is, the upper surface of the airfoil is half the thickness above the mean camber line, and the lower is half the thickness below.*

$$z_u = \bar{z} + \frac{t}{2}$$
$$z_\ell = \bar{z} - \frac{t}{2}$$

Create a function that takes whole number values for max camber, c; max camber position, p; max thickness, t; and an array of x values spanning (0, 1) (hint, use the range function). The function should take the whole numbers for c, p, and t, and convert them into the correct magnitudes (c and t are in units chord/100, and p is in units chord/10). The function should call your thickness and camber functions and then solve for the upper and lower z coordinates at each x coordinate, and return both of the z vectors.

If you try and call your thickness function as it currently is with the entire x array as an input, you're probably going to get some errors (you'll have an issue with the square root function, most likely.) While you could use a loop to call your thickness and camber functions and solve for the upper and lower z coordinates for each x coordinate one at a time to avoid the issue, it is much simpler to use broadcasting. This topic is slightly advanced, but learning good practices now will pay off later. Look through the documentation on broadcasting just linked. Then try running your thickness function as thickness.() (with the dot after the function name).

To check your work, look at a NACA 2412 airfoil, that is to say, input $c = 2$, $p = 4$, $t = 12$ and x = [0.0; 0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 1.0], and recreate the following table.

x	z_u	z_ℓ
0.0	0.0	0.0
0.1	0.055565104238239515	-0.038065104238239514
0.2	0.0723250299023625	-0.0423250299023625
0.3	0.07865386639397029	-0.04115386639397029
0.4	0.07782850847647903	-0.03782850847647902
0.5	0.07206969644501603	-0.03318080755612714
0.6	0.06295786843645562	-0.02740231288090007
0.7	0.051021667126780204	-0.021021667126780205
0.8	0.03653589091583613	-0.014313668693613908
0.9	0.01956768382582959	-0.00734546160360737
1.0	-1.6653345369377347e-17	1.6653345369377347e-17

You should also practice accessing specific elements of your output array. Try extracting arbitrary ranges of the arrays, or grabbing full rows or columns. When you are comfortable with that, do the following:

- Copy the second and third columns into their own 1D arrays.

- Concatenate the z arrays to create a single 1D array that starts with the last element of z_ℓ and ends with the last element of z_u. Do the same for the x-values such that you have an array that goes from 1 to zero to 1.