MIE 124
Homework 5

For your homework, turn in two m_files, called hw5_yourinitials_func.m and hw5_yourinitials_script.m. One should be a function, and the other should be a script file, respectively. You will be playing a game called Mastermind.

For the game "Mastermind", you will create a function that simulates the game between two players. In the game, the first player is the Code Maker, who creates a certain code selected from the available colors. So for example, a three digit code could be: green, yellow, blue. The second player is the Code Breaker, who is trying to guess the code made by the Code Maker. Each round, the Code Breaker makes a guess, and the Code Maker returns information regarding the guess. The Code Maker returns the number of correct colors that are in the correct spot, but does not reveal which colors are in the right spot. The Code Breaker then guesses again. The game continues until the Code Breaker guesses the right code or until 8 guesses have been made.

Another example: if there are 6 possible colors (red, yellow, green, blue, white, black) and 4 digits, the Code Maker could create the following code in this order: yellow, green, white, blue. NO COLORS MAY BE REPEATED. The Code Breaker does not know what the code is, so they will make their first guess, which could be: green, blue, white, red. The Code Maker would then return 1, because there is only one correct color in the correct spot. In this case, white was correctly placed. The Code Breaker would then guess a new combination in an attempt to match the Code Maker's code, and the Code Maker would return the new information.

For your function, you will represent the colors as numbers. For example, if there are 6 possible colors (red, yellow, green, blue, white, black), red = 1, yellow = 2, green = 3, blue = 4, white = 5, black = 6. So, in the previous example, the code would be equal to 2354. The Code Breaker's first guess would be equal to 3451. NOTE: YOU CANNOT HAVE MORE THAN 9 OR LESS THAN 1 COLORS.

<u>YOUR FUNCTION</u> - hw5_yourinitials_func.m
Create a function with two inputs: <u>number of colors (C)</u> and <u>number of digits (D)</u>, and one output: <u>number of rounds of guesses (R)</u>.

1) Using the provided function named gen_vector.m, create a vector called *guess_vector* that stores all possible color combinations given the digits. Do not alter the gen_vector.m function. The inputs to gen_vector are C and D, and the output is a vector with all possible codes given the values of C and D. So if 6=4 and D=4, then there are 360 possible codes.
2) Randomly select a single value from *guess_vector*, and assign it to the variable *answer.* This is the code your function will guess.
3) Now randomly select an initial guess from *guess_vector*. Call this variable *guess*.

4) Initialize a logical vector called *TF*. It should be the same size as *guess_vector*, and should be initialized as all true. This will help when eliminating guesses later.
5) Initialize R with a value of 1.
6) Now it's time to play the game. Use a while loop that runs while *guess* and *answer* are NOT equal and while the number of rounds is less than or equal to 8. For each iteration of the while loop:
    a. Use a subfunction called *check_matches* with two inputs (n1,n2) and one output (N_match). The inputs are just two integer numerical values, and the output is the number of matches in the digits between the two numbers. Using the example above, if you call *check_matches*(2354,3451), then the output should be N_match=1. HINT: You can break a number up into its individual digits using the following line of code:

        dig = dec2base(n,10)-'0';

        n is a number, and dig is a vector in which each element is a digit of n. So if n=1234, then dig = [1 2 3 4].

    b. Given the output of *check_matches*, you can eliminate many possibilities from your *guess_vector*. You should eliminate any number in the *guess_vector* that **does not have the same number of matches with *guess* as the matches returned from *check_matches***. For example, if you guess 1234 and *check_matches* returns 1, you will eliminate any number that doesn't have at exactly one match with 1234 (ex: 2341 [no matches], 1243 [2 matches], 1235 [3 matches], etc.) Use a for loop to compare each value in guess_vector to guess and change the corresponding element in TF to false if the number of matches is not correct.
    c. Use your *TF* vector to eliminate guesses from *guess_vector*. *guess_vector* will shrink in size with each round.
    d. Select a new guess from the updated *guess_vector*. Increment the value of R by 1, and update TF so that it is the same size as guess_vector and has all true values once again. Repeat the steps to continue the game, which will end once *guess* is equal to *answer* or if *R* exceeds 8.

<u>YOUR SCRIPT</u> – hw5_yourinitials_script.m
Create a script that will examine the change that occurs with different combinations of C and D.

1) Clear all variables and shuffle the seed.
2) Create a 3x2 matrix of inputs, with C in the first column and D in the second column. There are three combinations of inputs: C = 6 and D = 4; C = 8 and D = 5; and C = 9 and D = 6.
3) Use two nested for loops to test each combination N times. Store the output of your function in a 3xN matrix called *R_all*, with each row corresponding to each of the three

input combinations, and each column is one realization of playing the game. Remember the output of your function is the number of rounds played to guess the answer. When your code is done, set N=20. When testing though, make it a smaller value otherwise it will take too long.

4) Create three separate histograms for the values of R for each combination of inputs. Use subplot to put them on one figure, and use titles and labels to make it clear which is which.
5) Create a 3x1 vector called win_perc, which is the win percentage for each of the three input combinations. Reminder: a WIN is when the Code Breaker guesses the code in 8 or less rounds. Anything greater is a LOSS.
6) Compute the average number of rounds it takes for each combination of C and D. Store this in a 3x1 vector called R_avg.

Notes:

- Your script may take a while to run, as it requires a lot of looping.
- Make sure you label your graphs correctly.
- Since your code may take a while to run, use your debugger to run separate sections of your code.
- To test your code, use something like C=3 and D=2 and use the debugger to walk through the code line by line. There are only 6 combinations in this case, so it's easy to see if your logic is correct and if the loops are working.