CS1112 Exercise 6 Name: _____ NetID: _____

You have until *Sunday, 10/2, at 9pm* to complete this exercise. You must get problems 1 and 2 checked off in discussion section, office hours, or consultant hours. Submit Problems 3, 4, and 5 using MATLAB *Grader*.

1 Different ways to create vectors

Type the following expressions in the MATLAB *Command Window* to see what kind of vectors they create. Write the resulting vectors (and answer the questions) on the blanks.

```
a = zeros(1,4)  %_____________

b = zeros(4,1)  %_____________  What do the arguments specify?_____________________

c = ones(1,3)   %_____________

d = 10:2:17     %_____________

f = 10:-1:17    %_____________

g = linspace(10,19,4) %_____________

h = linspace(10,6,5)  %_____________

k = [10 20 40]  %_____________  What does the space separator do?_________________

m = [10,20,40]  %_____________  What does the comma separator do?_________________

n = [10;20;40]  %_____________  What does the semi-colon separator do?_____________

p = [a k]       %_____________

q = [b; n]      %_____________

r = [a n]       %ERROR--mismatched dimensions! (Attempt to concatenate a column to a row)

s = b'          %_____________  This operation is called "transpose"

t = [a b']      %_____________
```

2 Check out the MATLAB debugger

In this problem, we will explore the use of the MATLAB debugger (a useful tool for finding errors in your code). Type the following code into a script called `onlyEven.m`:

```
% Generate random ints in [0, 1, ..., 10] and store the generated number if it is even.
% Stop when 0 is generated and do not store 0.
k = 0;  % vector length so far
maxNum = 10;
num = ceil(rand*(maxNum+2))-1;
v = [];   % initialize empty vector
while num>0
    if  rem(num,2)==0
        k = k+1;
        v(k) = num;
    end
    num = ceil(rand*(maxNum+2))-1;
end
```

It generates random integers in [0, 1, ..., 10] and stores only the even integers that are generated. The code stops when the randomly generated number is 0.

(a) Add a **breakpoint** to the line where `k` is incremented (to add a breakpoint, click on the line number [for newer versions of MATLAB] or click on the dash next to the line number [for older versions of MATLAB]), then run the script. When it pauses, you will see a green arrow next to that line. Look at the value of `k` in the Workspace; did MATLAB pause *before* or *after* executing that line? [ ]

(b) **Step** the program once; the green arrow should move next to the `end` keyword. What value does `k` have now? [ ]

(c) **Continue** the program; it should stop at your breakpoint again. Which variables have changed?
[ ]

(d) There is an error in the code. Create a breakpoint in the one or both of the `num = ceil(rand*(maxNum+2))-1;` lines and try to identify the error by stepping through the code. You may have to do this a few times to find the error. What is the error? [ ]

3 Basic loop pattern for a vector

See MATLAB Grader

The objective of this problem is to give you practice with basic loop patterns on vector—do not try to circumvent the practice by using built-in functions.

(a) Accumulation Pattern: Compute the sum of all the elements in vector `v`. Do not use built-in function `sum`.

(b) Finding the best in a set: Find the maximum value in vector `v`. Do not use built-in functions `max`, `min` and `sort`.

4 Searching within a vector

See MATLAB Grader

Write a function `vectorQuery(v,n,r)` to determine whether the number `r` appears in the first `n` components of vector `v`. The function returns `true` if `r` is in the first `n` components of `v` and `false` otherwise. Assume that `r` is an integer, `v` stores integer values, and `n` is a positive integer. Make effective use of a loop to do the search. Do not use any built-in functions other than `length`, `min`, `max`. Do not use vectorized code. Make sure that the loop index doesn't go "out of bounds" (if `n` is greater than the length of vector `v`). *Be efficient*: the loop should stop as soon as `r` is found.

5 Creating vectors of unknown length

See MATLAB Grader

Write a function `sequence(m)` that generates a sequence of random *integer* numbers between 1 and m, inclusive, stopping when a value is repeated for the first time. $m > 1$. The function returns a vector containing all the numbers generated (in the order in which they were generated) except for the last value that is a repeated occurrence.

Example: If the generated sequence is 3 1 9 5 7 2 5, the vector to be returned should be 3 1 9 5 7 2.

Notes: 1) Use built-in functions `rand`, `floor`, `ceil` to generate random integer values; *do not* use function `randi`. 2) Use a `while`-loop since this problem is a case of indefinite iteration—the number of iterations needed is not known in advance. 3) Make effective use of the function `vectorQuery` that you have developed already—*Do not* use built-in functions `find` or `contains`. 4) When you don't know how long a vector needs to be, you can build it one component at a time. We build a vector one component at a time in `onlyEven.m` so please review that script if you are unsure of how to do this.