CS1112 Exercise 10

You have until *Sunday, 10/30, at 9pm* to complete the exercise. **Get Problem 1 checked off** by your TA or go to consultant hours/TA hours to get this problem checked off. **Submit your solutions to Problems 2 and 3 on MATLAB Grader**. **Problem 4 is optional**–do it if you want to get ahead and test out cell arrays! You may want to develop your code using the full MATLAB environment first, taking advantage of MATLAB debugging tools, before using MATLAB Grader for further testing and submission.

1 `char` arrays

We are using the type `char` in MATLAB, not `string`. What we will call a char array is really an *array of* `char`s. Type each of the following statements in the *Command Window* and note the result.

```
a = pi;   % A numeric scalar
b = 'pi'  % A char array. Use SINGLE quotes to enclose a char or multiple chars

c = length(b)            % __________  b is an array, so one can use function length on it

d = ['apple '  b  'es']  % Vector concatenation. d should be the string 'apple pies'

e = [d; 'muffin']        % ________________________________________

e = [d; 'mmmuffins ']    % Note the two extra 'm's and one trailing space

[nr,nc] = size(e)        % ______________ e is a matrix, so one can use function size on it

f = e(1, 7:9)            % ________________________________________ Accessing a subarray

d(1, 7:10) = 'core'      % ________________________________________

h = 'P' - 'M';           % __________ How many letters is 'P' after 'M'?

ii = char('d' + 4)       % __________ What comes 4 letters after 'd'?

jj = ii>='A' && ii<='Z'  % __________ True or false: character stored in ii is upper case

k =  ii>='a' && ii<='z'  % __________ True or false: character stored in ii is lower case

L = strcmp('abcd', 'ab') % _________________________________ strcmp compares the arguments

m = 'abcd'=='ab'         % ERROR: attempted vectorized code on vectors of different lengths

n = 'abcd'=='abCd'       % ____________________________ Vectorized code--result is a VECTOR

o = sum('abcd'=='abCd')  % ____________________________ The number of matches

n = sum('abcd'~='abCd')  % ____________________________ The number of mismatches
```

2 Reverse complement

In the DNA double helix, two strands twist together and "face" each other. The two strands are reverse-complementary, i.e., reading one strand in reverse order and exchanging each base with its complement gives the other strand. A and T are complementary; C and G are complementary.

For example, given the DNA sequence	`AGTAGCAT`
the reverse sequence is	`TACGATGA`
so the reverse complement is	`ATGCTACT`

(a) Write a function `rComplement(dna)` to return the reverse complement of a DNA strand. *Use a for-loop* to reverse the strand—do not use vectorized code. `dna` is a vector of characters. Assume that `dna` contains only the letters 'A', 'T', 'C', and 'G'. If `dna` is the empty vector, then return the empty vector. Do *not* use any built-in functions other than `length` and `size`.

(b) Write a function `rCompBulk(mat)` to return the reverse complements of a set of DNA strands. `mat` is a matrix of characters; each row of the matrix represents one strand of DNA (so `mat` contains only the letters 'A', 'T', 'C', and 'G'). Return a matrix the same size as `mat` such that the rth row of the returned matrix is the reverse complement of the rth strand of DNA (the rth row of `mat`). Again *use for-loops*—do not use vectorized code. In order to get practice with 2-d `char` array syntax, write code to work on 2-d array `mat` directly—do not call function `rComplement` from part (a) above.

3 Counting a DNA pattern

Write a function `countPattern(dna,p)` that returns how many times a pattern `p` occurs in `dna`. Assume both parameters to be `char` vectors that contain the letters 'A', 'T', 'C', and 'G' only. Note that if `p` is longer than `dna`, then `p` appears in `dna` zero times. Use a `for`-loop to solve this problem.

(a) Version 1: Use the built-in function `strcmp` to compare two `char` vectors.

(b) Version 2: Do not use `strcmp`; instead use vectorized code and `sum` as demonstrated in Part 1 above to compare two `char` vectors.

4 Cell array vs. vector

You already know that a vector is a collection of simple data. For example, you can have a vector of numbers (each component stores *a single number*) or a vector of characters (each component stores *a single character*). In a cell array, each cell can store an item that may be more complex than just a number or a character.

Type the following code in the command window and observe the output and the display in the *Workspace* pane.

```
v = rand(1,4)  % a VECTOR of length four, each element stores ONE number
v(3)           % Notice that you use PARENTHESES to access a cell in a VECTOR

c = cell(1,4)  % c is a CELL ARRAY.  c's "class" in the Workspace pane is "cell."
               % Right now each cell has an empty vector.

c{2} = v       % Put a VECTOR inside the 2nd cell of the CELL ARRAY.  Notice that we use
               %   CURLY BRACKETS to access contents of a cell in a CELL ARRAY.

c(3) = 1       % Error: cannot change the type cell (left side) to type double (right side).
               %   c(3) gets the cell, not the contents INSIDE the cell.  Must use curly
               %   brackets to access the contents inside a cell of a CELL ARRAY.

c{2}           % Display what is inside cell 2 of CELL ARRAY c:  a vector!

c(2)           % Display the cell (the "wrapper"), not what is inside the cell

% So how do you display, say, the fourth value in the VECTOR in the 2nd cell of CELL ARRAY c?
c{2}(4)        % Once again, use curly brackets for the index of the CELL ARRAY; use
               %   parentheses for the index of the of VECTOR.

c{1} = 'cat'   % OK for individual cells of a cell array to have different types
c{3} = 10
c{4} = ones(2,1)

% An alternate way to create a cell array is to specify all the contents inside CURLY
%   BRACKETS using spaces, commas, or semi-colons as the separator:
d = {'cat'; 10; v; ones(2,1)}  % A cell array of four cells
e = length(d)                  % The length function works for cell arrays as well.
```