CS1112 Lab Exercise 10

1.1 Not string but `chars`

In MATLAB, there is the type `char` but not the type string. What we call a string 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

e(1, 7:10)= 'core'       % _________________________________________

g= ones(2,3)*67;         % A NUMERIC 2-by-3 matrix, each component has the value 67

h= char(g)               % _________________________________________

i= double(h)             % _________________________________________

jj= char(floor(rand*26) + 'A')  % ___________________________ A random upper case letter

k= jj>'a' && jj<'z'      % __________ True or false: character stored in jj 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
```

1.2 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 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 return the empty vector.

(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 loops*—do not use vectorized code.

2.1 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. Also read the comments given below.

```
v= rand(1,4)  % a VECTOR of length four, each cell 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 in the 2nd cell of the CELL ARRAY.  Notice that we use CURLY
              %   BRACKETS to access a cell in a CELL ARRAY.

c(3)= 1       % Error: Must use curly brackets to access a cell in a CELL ARRAY;
              %   parentheses are for VECTORS.

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

% 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.
```

2.2 Deck of cards

Download the functions `CardDeck` and `Shuffle` from the *Lecture Materials* page. Read the code and run the functions to make sure that you understand them. Ask if you have questions. Implement the following functions as specified:

```
function DispCards(ca, p, q)
% Display the contents in cells p through q of cell array ca.
% ca is a 1-d cell array.

function sd= MyShuffle(d)
% d is a one-dimensional cell array
% sd is the cell array after shuffling d
% The shuffle comprises two steps:
%  - randomly cut the deck into 2 parts. I.e., the position of the cut is random.
%  - interleave the cards from the two parts until the part with fewer
%    cards have been completely incorporated.  It is up to you whether
%    to start from the top or the bottom.
```