CS1112 Exercise 11 Name: ______________________ NetID: ________

You have until *Sunday, 11/6, at 9pm* to complete the exercise and submit Questions 2 and 3 on MATLAB Grader. No in-person check-off is necessary, but asking questions in-person is strongly encouraged! 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. Question 1 is important for understanding the different syntax associated with arrays but does not need to be submitted.

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 an element in a VECTOR

c = cell(1,4) % Use built-in function CELL to create a CELL ARRAY.  Note that its "class" in
              %   the Workspace pane is "cell."  Right now each cell is empty, therefore the
              %   screen output shows four empty vectors.

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      % You get an error message: Use curly brackets to access a cell in a CELL ARRAY;
              %   use parentheses for VECTORS, or some cases of vectorized code (see syntax
              %   details posted with Lecture 18 notes).

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.
```

2 Deck of cards

Download the functions `CardDeck`, `Shuffle`, and `DispCards` from the course website. Read the code and try calling the functions to make sure that you understand them. Ask if you have questions. Implement the following function as specified:

```
function [c, sd]= MyShuffle(d)
% d is a one-dimensional cell array representing a deck of cards.
% sd is the cell array after shuffling d.
% c is the "cut point," explained below.
% The shuffle comprises two steps:
%  - Cut the deck into two parts:  generate a random integer value such that it is equally
%    likely to be any value in the set of numbers [15, ..., 38].  This value is c, the first
%    return parameter. The Top subdeck gets the cards at positions 1 to c of the original deck;
%    the Bot subdeck gets the cards at positions c+1 to n of the original deck.
%  - Interleave the cards from Top and Bot subdecks in this order:
%    Top{1}, Bot{1}, Top{2}, Bot{2}, ..., Top{m}, Bot{m}, where m is the smaller value
%    between c and 52-c.  (I.e., interleave the cards from the two subdecks until the
%    smaller subdeck has been completely incorporated.)  Then put any remaining cards
%    (from the larger subdeck) at the end of the shuffled deck.
% Use built-in functions rand and floor (or ceil) for generating the cut point.
% Write NON-vectorized code.
```

Hint on generating a random cut point: Only one of these two expressions is correct: `floor(rand(1)*(38-15+1))+15` or `floor(rand(1)*(38-15))+15`. Do you know which one is correct? Why?

In order to practice indexing and cell array syntax, do not try to vectorize your code. I.e., work with one cell at a time. During code development, you can call the given `DispCards` *inside* the function `MyShuffle` to show the cell array being built in order to confirm that the intermediate steps are correct. Just remove the calls to `DispCards` after you've completed the code.

3 More card playing . . .

Implement the following function:

```
  function sd = Cut3(d)
  % d is a one-dimensional cell array whose length is a multiple of 4.
  % sd is the cell array after cutting the deck d by taking half the cards
  %   from the middle of the deck and putting that half on top.
```

During program development you can use `DispCards` to confirm that `Cut3` is implemented correctly, but `DispCards` should not be used in (the final implementation of) function `Cut3`.

Consider this example. Suppose deck `d` is 12 cards in this order:

Ace, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack

After calling function `Cut3` on this deck, the returned deck should be ordered as

3, 4, 5, 6, 7, 8, Ace, 1, 2, 9, 10, Jack

Challenge (no submission necessary): Write a script to find out whether the cards in the deck cycle back to the original arrangement after repeated cuts done by function `Cut3`. If so, how many cuts are needed to cycle back? You may use the function `strcmp` to compare two `char` vectors.