CS 1132 Fall 2016 Assignment 1a due 9/6 at 11:59 pm

Adhere to the Code of Academic Integrity. You may discuss background issues and general strategies with others and seek help from course staff, but the implementations that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is never OK for you to see or hear another student’s code and it is never OK to copy code from published/Internet sources. If you feel that you cannot complete the assignment on your own, seek help from the course staff.

When submitting your assignment, follow the instructions summarized in Section 3 of this document.

Do not use the break or continue statement in any homework or test in CS1132.

1 Egyptian Rat Slap

Egyptian Rat Slap is not actually about rats—it is a card game! However, we will not be simulating the game in its entirety; this assignment is merely inspired by it. Below are some rules of the game:

- To play, you need is a deck of 52 cards and at least two players.
- To win the game, you must collect all 52 cards.
- The dealer should split the 52 cards evenly among all players, face down. No one should look at their cards at any point of time.
- The person to the left of the dealer begins by flipping her card face up in the center of the table.
- The next person does the same, placing her card on top of the previous one. This should continue in a counterclockwise fashion around the table.
- Each person must be on the lookout for a “slap” combination, described in the next few sections. When one of these combinations appears, the fastest person to slap the pile of cards collects the cards and places them at the bottom of her own pile.
- If someone accidentally slaps the pile without a slap combination present, that person must give up two of her cards and place it at the bottom of the center pile.

There are a few extra rules, but we will not discuss them here. If you are interested in learning how to play the actual card game, you can watch this video: https://www.youtube.com/watch?v=1p0J0cJX48Y.

By the end of this assignment, you will be able to determine how many slap combinations are made within approximately one round of the game. For simplicity, we assume that one player is playing against himself and that he has great reflexes and the visual acuity to catch all of the slaps as they appear, without slapping incorrectly.

You will write several functions and complete the script assignment1a.m.

1.1 Creating a Deck

Implement the following function to create a standard deck of cards:

```matlab
function deck = deckCreator()
% deck is a shuffled deck of 52 cards, stored as a 1-D array of numbers.

Usually a deck of cards is represented by four suits (Hearts, Diamonds, Spades, and Clubs) and thirteen ranks (Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, and King). However, for Egyptian Rat Slap, we do not need to worry about the suits. Therefore, we will represent the cards using integers from 1 to 13, where Ace
is 1, Jack is 11, Queen is 12, and King is 13. These 13 integers are repeated four times to make a full set of 52 cards.

Start by initializing a vector with the values 1, 2, . . . , 13, 1, . . . , 13, 1, . . . , 13. Then “shuffle the deck” by repeatedly choosing two indices at random and swapping the values at those indices. The indices should be randomly generated such that

- one index is equally likely to be any integer in [1 . . . 26], and
- the other is equally likely to be any integer in [27 . . . 52].

Repeat the random generation of a pair of indices and swapping at least 100 times. You may use built-in function randi to generate a random integer. (Search the MATLAB documentation or type in the Command Window help randi or doc randi to learn how to use randi.\footnote{Instead of randi, one could use rand to generate a random real value and then convert it to an integer. E.g., to get a random integer in the range \([a \ldots b]\) where \(a\) and \(b\) are integer values and \(a < b\), one can use the expression \(\text{floor}(\text{rand} \ast (b-a+1))+a\), i.e., \(\text{floor}(\text{rand}+\text{number\_of\_possibilities})\ast\text{base\_value}\).}

After implementing the deckCreator function, add code to the assignment1a.m file to call the function and display the resulting deck of cards.

### 1.2 Doubles

Doubles are one of the slap combinations. A double has been played when two consecutive cards have the same rank. There are different slap combinations that can be played simultaneously during a game, but for this section, let’s assume we will only care about doubles.

Write a function doubles that counts the number of doubles that can be slapped given a deck of cards. The function takes a 1-D array of cards as input and returns the number of doubles found as output. You can assume that the input vector has a length greater than 2. Make sure to not count a card twice to give two (false) doubles. For example, these six cards of a deck

\[
7 \quad 11 \quad 11 \quad 11 \quad 1 \quad 1
\]

give only two doubles, the first two Jacks and the last two Aces. (Recall that given perfect game play, the first two Jacks would have been slapped and removed and so the third Jack is not part of a double.) On the other hand, these seven cards of a deck

\[
7 \quad 11 \quad 11 \quad 11 \quad 11 \quad 1 \quad 1
\]

give three doubles (the first two Jacks, the next two Jacks, and the last two Aces).

After writing function doubles, call the function in assignment1a.m on the deck given to you to test your work. Then create extra decks in the script file to make sure that the function works in other scenarios.

### 1.3 Sandwiches

Sandwiches are another type of slap combination. Three cards are needed to create a sandwich, where the first two cards have the same rank and the middle card is of a different rank. For example, in the cards below

\[
6 \quad 5 \quad 10 \quad 5 \quad 10 \quad 5 \quad 1
\]

The first 5 10 5 combo is a sandwich.

Write a function sandwiches that counts the number of sandwiches that can be slapped. The function takes a 1-D array of cards as input and returns the number of sandwiches as output. You can assume that the input vector has a length greater than 3. Note that your algorithm should be similar to the doubles function, but it will require slightly different bounds for the loop condition and the loop increment.

Once you have written the function, again call the function in assignment1a on the given deck and your extra decks to test your work.
1.4 Doubles AND Sandwiches

This time, write a function that checks for BOTH doubles and sandwiches that can be slapped in a game! Do not count a card as part of two combinations at a time. A card can be part of a double OR part of a sandwich, depending on which appears first. For example, the sequence of cards

\[5 \ 5 \ 10 \ 5 \ 1\]

gives only one double (not a double and a sandwich, and not a sandwich).

The function takes as input a 1-D array of cards and has two return parameters: one for the number of doubles found and the other for the number of sandwiches found. Use the following function header:

\[
\text{function } \text{[numDoubles, numSandwiches]} = \text{doublesAndSandwiches(deck)}
\]

Again, test your function in the assignment1a.m file.

2 Self-check list

The following is a list of the minimum necessary criteria that your assignment must meet in order to be considered satisfactory. Failure to satisfy any of these conditions will result in an immediate request to resubmit your assignment. Save yourself and the graders time and effort by going over it before submitting your assignment for the first time.

Note that, although all of these are necessary, meeting all of them might still not be sufficient to consider your submission satisfactory. We cannot list everything that could be possibly wrong with any particular assignment!

- Comment your code! If any of your functions is not properly commented, regarding function purpose and input/output arguments, you will be asked to resubmit.
- Suppress all unnecessary output by placing semicolons (;) appropriately. At the same time, make sure that all output that your program intentionally produces is formatted in a user-friendly way.
- Make sure your functions' names are exactly the ones we have specified, including case.
- Check that the number and order of input and output arguments for each of the functions matches exactly the specifications we have given.
- Test each one of your functions independently, whenever possible, or write short scripts to test them.
- Check that your scripts do not crash (i.e., end unexpectedly with an error message) or run into infinite loops. Check this by running each script several times in a row. Before each test run, you should type the commands `clear all; close all;` to delete all variables in the workspace and close all figure windows.

3 Submission instructions

1. Upload files deckCreator.m, doubles.m, sandwiches.m, doublesAndSandwiches.m and assignment1a.m to CMS in the submission area corresponding to Assignment 1a in CMS before the deadline.
2. When the scores are released read the grader’s feedback carefully.
3. If you need to resubmit, fix all the problems and go back to Step 1! Otherwise you are done with this assignment. Well done!