# 1 Description

The goal of this assignment is to shuffle and deal a deck of cards for a game of standard one-card deal Solitaire.

A standard deck of cards has a total of 52 cards with 13 cards to each suit. For the purpose of this project the Jokers will be ignored. Also an Ace will have a value of 1, a Jack will be 11, a Queen will be 12, and a King will be 13. In the demo you implemented a function to create a deck of cards. The function should create a vector of vectors, meaning each value of the vector should be a vector with values for the suit and the card value.

**Part 1**

The first part is to shuffle the deck that has just been created. The function should also return a vector of maps like the first function did. Note that there is a native shuffle function available, however you are not permitted to use this.



The layout of a game of Solitaire. When playing the game only the top cards of the 7 piles are visible. However, for this project you will not need to worry about visibility only about getting the correct number of cards into the correct number of piles, as explained below.

**Part 2**

The second part is to create a function to deal out the cards so that they can be used for gameplay. The second function will accept a vector of vectors as a parameter. The shuffle function should return a shuffled deck of cards. The deal function should return a nested vector of 8 piles - 7 piles for the playable cards and 1 pile for the rest of the deck. The first 7 piles should be of increasing size. That is that the first pile should have 1 card, the second pile 2 cards, the third pile 3 cards, etc. The final 8th pile should be the remainder of the deck. The ultimate goal of a standard game of solitaire is to move through the cards in the seven piles and the waste pile so that there is an ordered pile for each suit. Once you find the Ace cards for each suit you begin to pile the cards in order on top of their corresponding Aces.

Both functions should be run through the -main function.

|  |
| --- |
| (ns solitaire.core)  (def ranks (range 1 14)) ;;range of values  (def suits [:hearts :spades :clubs :diamonds]) ;; suits  (def deck (into [] (for[s suits r ranks] [s r]))) ;; deck of cards  (defn shuffle-deck [deck-of-cards] (...) )  (defn deal [deck-of-cards] (...) )  (defn -main[] (...) ) |

When creating a new project use this command:

lein new solitaire

# 2 Submission

Turn in a zip file containing all project files created by Leiningen. Your project will not run if the entire file structure is not included. Your code should also be well documented. Remember that a semicolon is used for comments.

# 3 Rubric

|  |  |  |
| --- | --- | --- |
| Project 3 - Clojure | | |
| Criteria | Ratings | Points |
| shuffle-deck[]   * The function correctly shuffles a deck of cards. * There are no repeated cards or missing cards. * The shuffled deck has 52 cards. * It accepts the specified parameters and returns the specified structure. * If you use the native shuffle function you will receive no credit for this function. |  | 15 pts |
| deal[]   * The function correctly sets up a game of Solitaire. * There are 8 piles and each pile has the correct number of cards. |  | 15 pts |
| -main[] - The main function is present and all other functions are run through it. |  | 10 pts |
| Documentation - The code is well documented. |  | 10 pts |
| Total Points: 50 | | |

# 4 Test Cases

shuffle-deck[] should accept a vector of vector as a parameter and shuffle it. Here is one possible correct outcome:

[[:hearts :1] [:hearts :8] [:clubs :6] [:spades :2] [:hearts :10] [:diamonds :11] [:clubs :4] [:hearts :7]

[:diamonds :6] [:spades :5] [:clubs :10] [:hearts :13] [:spades :11] [:clubs :5] [:clubs :11] [:diamonds :13] [:hearts :2] [:diamonds :7] [:spades :6] [:spades :12] [:hearts :3] [:diamonds :8] [:clubs :12] [:spades :7] [:diamonds :2] [:hearts :9] [:spades :13] [:spades :1] [:diamonds :3] [:clubs :13] [:clubs :1] [:hearts :4] [:spades :8] [:clubs :7] [:diamonds :9] [:hearts :5] [:diamonds :4] [:spades :3] [:spades :9] [:clubs :] [:hearts :11] [:diamonds :10] [:clubs :2][:hearts :6] [:hearts :12] [:clubs :9] [:spades :4] [:clubs :3]

[:diamonds :5] [:spades :10] ]

This next outcome for shuffle would be incorrect because card values are repeated and some of the cards are missing:

[[:hearts :1] [:hearts :8] [:clubs :6] [:spades :2] [:hearts :10] [:diamonds :11] [:clubs :4] [:hearts :7] **[:hearts :9]** [:spades :5] [:clubs :10] [:hearts :13] [:spades :11] [:clubs :5] [:clubs :11] [:diamonds :13] [:hearts :2] [:diamonds :7] [:spades :6] [:spades :12] [:hearts :3] [:diamonds :] [:clubs :12] [:spades :7] [:diamonds :2] **[:hearts :9]** [:spades :13] **[:spades :1]** [:diamonds :3] [:clubs :13] [:clubs :1] [:hearts :4] [:spades :] [:clubs :7] [:diamonds :9] [:hearts :5] **[:spades :1]** [:spades :3] [:spades :9] [:clubs :]

[:hearts :11] [:diamonds :10] [:clubs :2] [:hearts :6] [:hearts :12] [:clubs :9] [:spades :4] [:clubs :3] [:diamonds :5] [:spades :10] ]

Finally deal should produce a nested vector of vectors of vectors (triple nested vector). This is one correct output:

[ [[:hearts :1]]

[[:hearts :7] [:clubs :1]]

[[:diamonds :13] [:clubs :7] [:hearts :2]]

[[:clubs :3] [:clubs :11] [:hearts :12] [:diamonds :9]]

[[:clubs :5] [:diamonds :1] [:spades :2] [:diamonds :10] [:spades :]]

[[:hearts :5] [:spades :5] [:hearts :2] [:diamonds :7] [:spades :4] [:hearts :11]]

[[:spades :13] [:diamonds :6] [:hearts :13] [:diamonds :4] [:hearts :1] [:spades :12] [:diamonds :5]]

[[rest of the deck]]]