CSE 231 **Spring 2020**

Computer Project #10

Assignment Overview

This assignment focuses on the design, implementation and testing of a Python program which uses an instructor-supplied module to play a card game, as described below.

It is worth 60 points (6% of course grade) and must be completed no later than 11:59 PM on Thursday, **April 16, 2020**.

Assignment Deliverables

The deliverable for this assignment is the following file:

`proj10.py` – the source code for your Python program

Be sure to use the specified file name and to submit it for grading via the **Mimir system** before the project deadline.

Assignment Background

Aces Up is a popular solitaire card game which is played by one person with a standard 52-card deck of cards. The rules and a tutorial video are available at:

http://worldofsolitaire.com/

Click on "Choose Game" at the top of the screen and click on "Montana".

Your program will allow the user to play the game Montana, with the program managing the game. The game rules are given below.

Game Rules

1. Start: The game is played with one standard deck of 52 cards with aces removed (so there are 48 cards). The deck is shuffled, and all 48 cards are dealt face up into a *tableau* of four rows of thirteen places each. When the 48 cards are dealt there will be four empty spaces.

```
Montana Solitaire.
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  4♣  2♠  6♦  9♣      2♣  7♥  9♠  4♥  7♣  4♠  7♠
  2:  6♣  2♥ 10♠  6♥  4♦  K♦  8♠  8♥  2♦  K♥  5♣  J♥  9♥
  3:  5♦  Q♥      K♣  7♦  3♠  3♥  5♠  9♦  J♠ 10♣  K♠  8♣
  4:  3♣  Q♦  J♦  6♠      8♦  5♥  J♣  Q♠  Q♣  3♦ 10♦ 10♥
```

(Implementation note: we won't actually remove the aces from the deck. Instead we use the aces as the place holders for the blank spaces. When we display the cards we don't display anything for an ace so it shows up as an empty space. Also, when we check whether a space is empty or has a card we can check whether the place in the tableau has an ace.)

Most solitaire games also have a *stock* and a *foundation* but not this game.

2. Goal: The game is won when each row of the tableau is an increasing sequence of cards of the same suit starting with a 2 in the leftmost column up through a king in the next to last column with a blank space in the rightmost column (actually holding an ace that isn't displayed, but the ace may be of any suit since it is simply representing a blank space). Note that any suit may be in any row, but the entire row must be of the same suit.

3. Moves: A player can move only one card at a time and only to an empty space. A move is valid only if the card to the left of the empty space is the same suit and one less in rank than the card being moved.

4. Shuffle: (Note that this is the hardest part of the game and should only be implemented after the rest of the game is complete.) Only some cards are shuffled: cards that are already in a same-suit sequence starting with a 2 in the leftmost column are left in place and not shuffled. All other cards are shuffled and then dealt back into the tableau, but there will be one blank space in each row and it will be at the right end of the sequence that is left in place (or in the leftmost column, if there is no sequence in that row).

Before shuffle with sequences highlighted in red

	1	2	3	4	5	6	7	8	9	10	11	12	13
1:	2♣	3♣	4♣	5♣	6♣		Q♣	J♠	9♠	8♠	10♥	6♠	8♣
2:		2♠	3♠	4♠	8♥	4♥	7♥	6♥	9♣	10♦	K♥	Q♥	J♣
3:	2♦	3♦	4♦	5♦	6♦	7♦	9♦	Q♠	5♥		K♣		J♦
4:	2♥	J♥	8♦	Q♦	K♠	10♣	3♥	7♠	7♣	10♠	K♦	9♥	5♠

After shuffle with sequences highlighted in red. Note the blank space to the right of each sequence (and in the leftmost column for a row with no sequence).

	1	2	3	4	5	6	7	8	9	10	11	12	13
1:	2♣	3♣	4♣	5♣	6♣		8♥	3♥	9♠	4♥	10♣	2♠	7♥
2:		5♠	5♥	6♥	10♦	Q♦	Q♣	8♣	8♦	K♦	J♥	K♠	10♥
3:	2♦	3♦	4♦	5♦	6♦	7♦		K♥	J♠	10♠	J♦	Q♥	K♣
4:	2♥		Q♠	7♣	3♠	9♥	4♠	6♠	J♣	9♣	8♠	7♠	9♦

Assignment Specifications

You will develop a program that allows the user to play Montana according to the rules given above. The program will use the instructor-supplied **`cards.py`** module to model the cards and deck of cards. To help clarify the specifications, we provide a sample interaction with a program satisfying the specifications at the end of this document.

1. The game will recognize the following commands (upper or lower case):

Sr Sc Dr Dc	Source row & column; Destination row & column; all ints
S	Shuffle Tableau (leaving sequences in place)
Q	Quit

where **Sr** and **Dr** denote row numbers (1 to 4); **Sc** and **Dc** denote column numbers (1 to 13).

The program will repeatedly display the current state of the game and prompt the user to enter a command until the user wins the game or enters "**Q**" (or "**q**"), whichever comes first.

The program will detect, report and recover from invalid commands. The tableau will not be altered by an invalid command.

2. The program will use the following function to initialize a game:

initialize() → tableau

That function has no parameters. It creates, initializes, and returns the tableau. This corresponds to the setup in the game rules:

- `tableau` is a list of 4 lists, each list containing 13 cards or empty spaces (aces). The first element of `tableau` is the leftmost column when displayed.

All cards are then dealt from the deck, left to right, one to each column of a thirteen-column `tableau` filling the top row first.

Note: we will use the whole deck of cards considering aces as empty spaces when displayed.

3. The program will use the following function to display the current state of the game:

display(tableau) → None

Provided.

Note: aces are displayed as blank spaces.

4. The program will use the following function to determine if a requested move is valid:

validate_move(tableau,source_row,source_col,dest_row,dest_col) → Bool

That function has five parameters: the data structure representing the tableau and four **ints** the source row & column and the destination row & column. Row and column `ints` are in the ranges 0 <= row <= 3 and 0 <= column <= 12. The function will return **True,** if the move is valid; and **False**, otherwise. A move is valid

- if the destination is empty (contains an ace)
- if the empty destination is the leftmost column and the source card has rank 2
- if the empty destination is not the leftmost column and the card to the left is the same suit as the source card and has a rank that is one less than the source card's rank.

Hint: this kind of function is easiest to code with many return statements, sometimes returning True and sometimes returning False. If some condition indicates that the move is valid or not, immediately return. For example, if the destination isn't empty (an ace), immediately return False. Another way to think of it is to look for ways to return False and at the end have a "return True" if you got that far without returning False.

5. The program will use the following function to move a card within the tableau:

`move(tableau,source_row,source_col,dest_row,dest_col) → Bool`

That function has five parameters: the data structure representing the tableau and four **`ints`** the source row & column and the destination row & column. Row and column `ints` are in the ranges 0 <= row <= 3 and 0 <= column <= 12. If the move is valid, the function will update the tableau and return True; otherwise, it will do nothing to it and return False.

Hint: swap the source card with the destination card (an ace) so the source now becomes an empty space.

6. The program will use the following function to check if the game has been won:

`check_win( tableau ) → Bool`

That function has one parameter: the data structure representing the tableau. The game is won when each row of the tableau is an increasing sequence of cards of the same suit starting with a 2 in the leftmost column up through a king in the next to last column with a blank space in the rightmost column (actually holding an ace that isn't displayed, but the ace may be of any suit since it is simply representing a blank space). Note that any suit may be in any row, but the entire row must be of the same suit.

Hint: this kind of function is easiest to code with many return statements, generally returning False if some condition isn't met and returning True at the end, if there was never a False returned.

7. The program will use the following function to shuffle cards in the tableau

`shuffle_tableau( tableau ) → None`

(Note that this is the hardest part of the game and should only be implemented after the rest of the game is complete.)

Only some cards are shuffled: cards that are already in a same-suit sequence starting with a 2 in the leftmost column are left in place and not shuffled. All other cards are shuffled and then dealt back into the tableau, but there will be one blank space in each row and it will be at the right end of the sequence that is left in place (or in the leftmost column, if there is no sequence in that row). See example above in the description of the game.
Hint: First, you need to extract all the cards that needs to be shuffled including the aces and store them into a list. Then, shuffle the list. After that, remove all the aces from the list and distribute the aces into the tableau. Last step is to distribute the remaining cards into the tableau.

11. Once you write all your function, it is time to write your main function:
    a) Your program should start by initializing the tableau.
    b) Display the tableau.
    c) Ask to input an option and check the validity of the input.
    d) If `'Q'`, quit the game
    e) If `'S'`, shuffle the tableau and display the tableau
    f) If **`'Sr Sc Dr Dc'`**, move card from Tableau (Sr, Sc) to empty Tableau (Dr, Dc).
    g) If none of these options, the program should display an error message.
    h) The program should repeat until the user won or quit the game.
    i) Then ask if the user wants another game
    j) Display a goodbye message.

Assignment Notes

1. Before you begin to write any code, play with the provided demo program and look over the sample interaction supplied on the project website to be sure you understand the rules of the game and how you will simulate the demo program. The demo program is at http://worldofsolitaire.com/: Click on "Choose Game" at the top of the screen and click on "Montana".

2. We provide a module called **`cards.py`** that contains a Card class and a Deck class. Your program must use this module (**`import cards`**). *Do not modify this file!* This is a generic module for any card game and happens to be implemented with "aces low" (having a rank of 1). Your game has blank spaces which we implement underneath with aces. Your program must implement that *without modifying the* **`cards`** *module***.**

3. Laboratory Exercise #11 demonstrates how to use the **`cards`** module. Understanding those programs should give you a good idea how you can use the module in your game.

4. We have provided a framework named **`proj10.py`** to get you started.
Using this framework is mandatory. Begin by downloading **`proj10.py`**. Check that it compiles. Gradually replace the "stub" code (marked with comments) with your own code. (Delete the stub code.)

5. The coding standard for CSE 231 is posted on the course website:

http://www.cse.msu.edu/~cse231/General/coding.standard.html

Items 1-9 of the Coding Standard will be enforced for this project.

6. Your program may not use any global variables inside of functions. That is, all variables used in a function body must belong to the function's local name space. The only global references will be to functions and constants.

7. Your program may not use any nested functions. That is, you may not nest the definition of a function inside another function definition.

8. Your program must contain the functions listed above; you may develop additional functions, as appropriate.

TEST 1 (no shuffle)

```
Montana Solitaire.
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣      5♥  Q♥  J♦
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦      K♥  8♥  6♥
  4:  2♥      Q♦  J♠ 10♥  9♥  8♠  Q♠  3♥  K♦  Q♣  J♥  J♣

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 13 1 10
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  J♦
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦      K♥  8♥  6♥
  4:  2♥      Q♦  J♠ 10♥  9♥  8♠  Q♠  3♥  K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 9 4 2
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  J♦
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦      K♥  8♥  6♥
  4:  2♥  3♥  Q♦  J♠ 10♥  9♥  8♠  Q♠      K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 1 13 3 10
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  K♥  8♥  6♥
  4:  2♥  3♥  Q♦  J♠ 10♥  9♥  8♠  Q♠      K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 3 11 1 13
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  K♥
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦      8♥  6♥
  4:  2♥  3♥  Q♦  J♠ 10♥  9♥  8♠  Q♠      K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 3 3 11
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  K♥
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥      J♠ 10♥  9♥  8♠  Q♠      K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 13 4 3
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  K♥
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥  4♥  J♠ 10♥  9♥  8♠  Q♠      K♦  Q♣  J♥
```

```
Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 12 2 6
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  9♠  K♠ 10♠  K♣
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥  4♥  J♠ 10♥  9♥  8♠  Q♠      K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 9 4 9
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  9♠     10♠  K♣
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥  4♥  J♠ 10♥  9♥  8♠  Q♠  K♠  K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 10 2 9
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  9♠ 10♠      K♣
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥  4♥  J♠ 10♥  9♥  8♠  Q♠  K♠  K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 4 2 10
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  9♠ 10♠  J♠  K♣
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥  4♥     10♥  9♥  8♠  Q♠  K♠  K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 1 11 4 4
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣      Q♥  K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  9♠ 10♠  J♠  K♣
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥  4♥  5♥ 10♥  9♥  8♠  Q♠  K♠  K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 11 1 11
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  Q♥  K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  9♠ 10♠  J♠  K♣
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥  4♥  5♥ 10♥  9♥  8♠  Q♠  K♠  K♦      J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 1 12 4 13
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣      K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  9♠ 10♠  J♠  K♣
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥  4♥  5♥ 10♥  9♥  8♠  Q♠  K♠  K♦      J♥  Q♥
```

```
Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 11 1 12
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  9♠ 10♠  J♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥  4♥  5♥ 10♥  9♥  8♠  Q♠  K♠  K♦      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 8 2 11
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  9♠ 10♠  J♠  Q♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥  4♥  5♥ 10♥  9♥  8♠      K♠  K♦      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 9 2 12
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  9♠ 10♠  J♠  Q♠  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥  4♥  5♥ 10♥  9♥  8♠          K♦      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 8 4 8
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥     10♠  J♠  Q♠  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥  4♥  5♥ 10♥  9♥  8♠  9♠      K♦      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 3 12 2 8
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  8♥ 10♠  J♠  Q♠  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦      6♥
  4:  2♥  3♥  4♥  5♥ 10♥  9♥  8♠  9♠      K♦      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 10 3 12
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  8♥ 10♠  J♠  Q♠  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦  6♥
  4:  2♥  3♥  4♥  5♥ 10♥  9♥  8♠  9♠              J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 9 4 9
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
  2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  8♥      J♠  Q♠  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦  6♥
  4:  2♥  3♥  4♥  5♥ 10♥  9♥  8♠  9♠ 10♠          J♥  Q♥
```

```
Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 6 2 9
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  8♥  9♥  J♠  Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦  6♥
 4:  2♥  3♥  4♥  5♥ 10♥      8♠  9♠ 10♠          J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 10 4 10
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  8♥  9♥      Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦  6♥
 4:  2♥  3♥  4♥  5♥ 10♥      8♠  9♠ 10♠  J♠      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 5 2 10
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  8♥  9♥ 10♥  Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦  6♥
 4:  2♥  3♥  4♥  5♥          8♠  9♠ 10♠  J♠      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 3 13 4 5
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  8♥  9♥ 10♥  Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥  5♥  6♥      8♠  9♠ 10♠  J♠      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 7 4 6
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠      8♥  9♥ 10♥  Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥  5♥  6♥  7♥  8♠  9♠ 10♠  J♠      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 7 2 7
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  8♥  9♥ 10♥  Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥  5♥  6♥  7♥      9♠ 10♠  J♠      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 8 4 7
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠      9♥ 10♥  Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥  9♠ 10♠  J♠      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 8 2 8
```

```
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠  9♥ 10♥  Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥     10♠  J♠      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 9 4 8
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠     10♥  Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥  9♥ 10♠  J♠      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 9 2 9
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠ 10♠ 10♥  Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥  9♥      J♠      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 10 4 9
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠ 10♠      Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥  9♥ 10♥  J♠      J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 10 2 10
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠ 10♠  J♠  Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥  9♥ 10♥          J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 12 4 10
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠ 10♠  J♠  Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥  9♥ 10♥  J♥          Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 13 4 11
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠ 10♠  J♠  Q♠  K♠
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥  9♥ 10♥  J♥  Q♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 1 13 4 12
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣
 2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠ 10♠  J♠  Q♠  K♠
```

```
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
  4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥  9♥ 10♥  J♥  Q♥  K♥
You won!

Do you want to play again (y/n)?n
Thank you for playing.
```

TEST 2 (with shuffle)

```
Montana Solitaire.
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣      5♥  Q♥  J♦
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦      K♥  8♥  6♥
  4:  2♥      Q♦  J♠ 10♥  9♥  8♠  Q♠  3♥  K♦  Q♣  J♥  J♣

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 13 1 10
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  J♦
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦      K♥  8♥  6♥
  4:  2♥      Q♦  J♠ 10♥  9♥  8♠  Q♠  3♥  K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 9 4 2
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  J♦
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦      K♥  8♥  6♥
  4:  2♥  3♥  Q♦  J♠ 10♥  9♥  8♠  Q♠      K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 1 13 3 10
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  K♥  8♥  6♥
  4:  2♥  3♥  Q♦  J♠ 10♥  9♥  8♠  Q♠      K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 3 11 1 13
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  K♥
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦      8♥  6♥
  4:  2♥  3♥  Q♦  J♠ 10♥  9♥  8♠  Q♠      K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 3 3 11
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  K♥
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥      J♠ 10♥  9♥  8♠  Q♠      K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 13 4 3
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  K♥
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
  4:  2♥  3♥  4♥  J♠ 10♥  9♥  8♠  Q♠      K♦  Q♣  J♥
```

```
Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 12 2 6
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  K♥
 2:  2♠  3♠  4♠  5♠  6♠  7♠  7♥  9♠  K♠ 10♠  K♣
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  8♥  6♥
 4:  2♥  3♥  4♥  J♠ 10♥  9♥  8♠  Q♠      K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: s
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣      7♥  Q♠
 2:  2♠  3♠  4♠  5♠  6♠  7♠     10♠  K♥  K♦  5♥  K♣  8♥
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦      8♠
 4:  2♥  3♥  4♥      Q♣  Q♥ 10♥  6♥  J♠  K♠  J♥  9♥  9♠

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 5 1 11
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  7♥  Q♠
 2:  2♠  3♠  4♠  5♠  6♠  7♠     10♠  K♥  K♦  5♥  K♣  8♥
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦      8♠
 4:  2♥  3♥  4♥          Q♥ 10♥  6♥  J♠  K♠  J♥  9♥  9♠

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 10 3 12
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  7♥  Q♠
 2:  2♠  3♠  4♠  5♠  6♠  7♠     10♠  K♥      5♥  K♣  8♥
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦  8♠
 4:  2♥  3♥  4♥          Q♥ 10♥  6♥  J♠  K♠  J♥  9♥  9♠

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 3 13 2 7
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  7♥  Q♠
 2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠ 10♠  K♥      5♥  K♣  8♥
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥          Q♥ 10♥  6♥  J♠  K♠  J♥  9♥  9♠

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 11 4 4
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  7♥  Q♠
 2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠ 10♠  K♥          K♣  8♥
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥  5♥      Q♥ 10♥  6♥  J♠  K♠  J♥  9♥  9♠

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 8 4 5
     1   2   3   4   5   6   7   8   9  10  11  12  13
 1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  7♥  Q♠
 2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠ 10♠  K♥          K♣  8♥
 3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
 4:  2♥  3♥  4♥  5♥  6♥  Q♥ 10♥      J♠  K♠  J♥  9♥  9♠

Enter choice:
```

```
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: s
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣      Q♠
  2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠      7♥  8♥  J♥  Q♥  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
  4:  2♥  3♥  4♥  5♥  6♥      9♠ 10♥  J♠  9♥  K♥ 10♠  K♣

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 7 2 8
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣      Q♠
  2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠  7♥  8♥  J♥  Q♥  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
  4:  2♥  3♥  4♥  5♥  6♥        10♥  J♠  9♥  K♥ 10♠  K♣

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 13 1 12
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  Q♠
  2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠  7♥  8♥  J♥  Q♥  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
  4:  2♥  3♥  4♥  5♥  6♥        10♥  J♠  9♥  K♥ 10♠

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 9 4 6
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  Q♠
  2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠      8♥  J♥  Q♥  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
  4:  2♥  3♥  4♥  5♥  6♥  7♥     10♥  J♠  9♥  K♥ 10♠

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 10 4 7
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  Q♠
  2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠          J♥  Q♥  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
  4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥ 10♥  J♠  9♥  K♥ 10♠

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 12 2 9
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  Q♠
  2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠ 10♠      J♥  Q♥  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
  4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥ 10♥  J♠  9♥  K♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 9 2 10
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  Q♠
  2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠ 10♠  J♠  J♥  Q♥  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
  4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥ 10♥      9♥  K♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 11 4 9
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣  Q♠
```

```
  2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠ 10♠  J♠      Q♥  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
  4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥ 10♥  J♥  9♥  K♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 1 13 2 11
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  Q♣  K♣
  2:  2♠  3♠  4♠  5♠  6♠  7♠  8♠  9♠ 10♠  J♠  Q♠  Q♥  K♠
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦  J♦  Q♦  K♦
  4:  2♥  3♥  4♥  5♥  6♥  7♥  8♥ 10♥  J♥  9♥  K♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: s
No more shuffles remain.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: q

Do you want to play again (y/n)?n
Thank you for playing.
```

TEST 3 (error checking)

```
Montana Solitaire.
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣      5♥  Q♥  J♦
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦      K♥  8♥  6♥
  4:  2♥      Q♦  J♠ 10♥  9♥  8♠  Q♠  3♥  K♦  Q♣  J♥  J♣

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col:
Error: invalid input.  Please try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: a
Error: invalid input.  Please try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 1 a 3 4
Error: invalid input.  Please try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 1.5 2 3 13
Error: invalid input.  Please try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2336
Error: invalid input.  Please try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 2 2 3
Error: invalid move.  Please try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 13 3 10
Error: invalid move.  Please try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 6 10 2 6
Error: row and/or column out of range. Please Try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 0 10 2 6
Error: row and/or column out of range. Please Try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 0 2 6
Error: row and/or column out of range. Please Try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 14 2 6
Error: row and/or column out of range. Please Try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 6 0 6
Error: row and/or column out of range. Please Try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 6 5 6
Error: row and/or column out of range. Please Try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 6 2 0
Error: row and/or column out of range. Please Try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 2 6 2 14
Error: row and/or column out of range. Please Try again.
```

```
Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 1 1 1 1
Error: invalid move.  Please try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 13 4 13
Error: invalid move.  Please try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: 4 13 1 10
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣  5♥  Q♥  J♦
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦      K♥  8♥  6♥
  4:  2♥      Q♦  J♠ 10♥  9♥  8♠  Q♠  3♥  K♦  Q♣  J♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: s
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣      K♠  9♥
  2:  2♠  3♠  4♠  5♠  6♠      Q♣  Q♠  K♣  Q♥  7♠  K♦ 10♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦      4♥  5♥ 10♠
  4:  2♥      8♠  J♦  J♠  Q♦  8♥  K♥  3♥  9♠  J♥  6♥  7♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: s
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣  J♣      Q♣  Q♥
  2:  2♠  3♠  4♠  5♠  6♠      Q♦  K♠  K♥  J♠ 10♥  8♠  7♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦      J♥  9♥  9♠
  4:  2♥      K♦  6♥  Q♠  3♥  7♠  J♦ 10♠  4♥  K♣  5♥  8♥

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: s
No more shuffles remain.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: a
Error: invalid input.  Please try again.

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: q

Do you want to play again (y/n)?y
Montana Solitaire.
      1   2   3   4   5   6   7   8   9  10  11  12  13
  1:  2♣  3♣  4♣  5♣  6♣  7♣  8♣  9♣ 10♣      5♥  Q♥  J♦
  2:  2♠  3♠  4♠  5♠  6♠      7♥  9♠  K♠ 10♠  K♣  7♠  4♥
  3:  2♦  3♦  4♦  5♦  6♦  7♦  8♦  9♦ 10♦      K♥  8♥  6♥
  4:  2♥      Q♦  J♠ 10♥  9♥  8♠  Q♠  3♥  K♦  Q♣  J♥  J♣

Enter choice:
 (q)uit, (s)huffle, or space-separated: source_row,source_col,dest_row,dest_col: q

Do you want to play again (y/n)?n
Thank you for playing.
```

Grading Rubric

Computer Project #10
Scoring Summary

General Requirements:
(5 pts) Coding Standard 1-9
(descriptive comments, mnemonic identifiers, format, etc…)

Implementations:

(7 pts) initialize
(8 pts) validate_move
(8 pts) move
    -3 for Boolean return
    -5 for Tableau update
(8 pts) check_win
(8 pts) shuffle_tableau
(6 pts) Test 1
(6 pts) Test 2
(4 pts) Test 3