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 65 points (5.5% of course grade) and must be completed no later than **11:59 PM on Monday, April 24, 2023**. After the due date, 1 pt will be deducted for every 1 hour late or a fraction of it. Note that you need to click on the submit button to be considered as submitted.

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 **Coding Rooms 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/

Under the "Solitaire" tab, click on "Select Game…" and "Aces Up".

Your program will allow the user to play a simplified version of Aces Up, 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. The deck is shuffled and becomes the initial *stock* (pile of cards). The cards in the stock are placed face down.

Four cards are then dealt from the stock, left to right, one to each column of a four-column *tableau*. During play, additional cards will be added to the tableau, but it starts with just one card in each column. All tableau cards are visible.

The game also has a *foundation*, which is a single pile of cards. The foundation starts out empty and is filled during play.

2. Goal: The game is won when the stock is empty and the only cards left in the tableau are the four aces.

3. Moves: A player can move only one card at a time and the moved card must be the bottom card of a tableau column.

The card at the bottom of a column can be moved to the foundation if a higher rank card of the same suit is at the bottom of another column. Note that aces are high in this game—that is, the rank of an ace is greater than the rank of any other card.

If a column of the tableau becomes empty, any card at the bottom of another column can be moved to the empty column.

No other moves are permitted.

4. Deal: A player can choose to deal instead of moving a card. In this case, the top four cards are dealt from the stock, left to right, with one card placed at the bottom of each column of the tableau. Note that the stock should never have fewer than four cards (there are 52 cards in the deck and you always deal four at a time).

Assignment Specifications

You will develop a program that allows the user to play Aces Up 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 on the project directory (**I_O_tests**).

The program will use the following functions to play the game:

`init_game() → (stock, tableau, foundation)`

a. The program will use this function to initialize a game. The function has no parameters and returns the starting state of the game with the three data structures initialized as described above.
b. Check the Game rules section to see how to deal cards to each data structure.

c. Parameters: no parameters
d. Returns: `stock (type Class Deck), tableau (list of lists), foundation (list).`
e. Display: nothing

`deal_to_tableau( tableau, stock ): → None`

a. The program will use this function to deal cards to the tableau.This function has two parameters: the data structure representing the stock and the data structure representing the tableau. It will deal a card from the stock to each column of the tableau, unless the stock has fewer than 4 cards; in which case it will just deal a card to consecutive columns until the stock is empty.

b. Parameters: `tableau (list of lists), stock (type Class Deck)`
c. Returns: nothing
d. Display: nothing

`validate_move_to_foundation( tableau, from_col ) → bool`

a. The program will use this function to determine if a requested move to the foundation is valid. The function should not modify the tableau in any case.
b. This function has two parameters: the data structure representing the tableau and an **`int`** indicating the index of the column whose bottom card should be moved.

c. The function will return **`True,`** if the move is valid; and **`False`**, otherwise. In the latter case, it will also print an appropriate error message. Check the game rules section to see when a move to foundation is valid.
d. Note that aces are high in this game—that is, the rank of an ace is greater than the rank of any other card. Also, you cannot move a card from an empty column.
e. `from_col` is the index of the column in the tableau (range from 0 to 3).
f. The error messages should look like this:

```
tableau  = [[ A♥], [], [ 2♦,  3♥], [ 4♠,  5♦,  6♦]]
validate_move_to_foundation(tableau,0) returns False
Error, cannot move  A♥.

validate_move_to_foundation(tableau,1) returns False
Error, empty column: 2

validate_move_to_foundation(tableau,2) returns True
```
tableau should not get modified.

g. Parameters: `tableau (list of lists), from_col (int)`
h. Returns: `bool`
i. Display: error messages

`move_to_foundation( tableau, foundation, from_col ) → None`

a. The program will use this function to move a card from the tableau to the foundation.
b. This function has three parameters: the data structure representing the tableau, the data structure representing the foundation, and an **`int`** indicating the index of the column whose bottom card should be moved.
c. If the move is valid, the function will update the tableau and foundation; otherwise, it will do nothing to them.
d. `from_col` is the index of the column in the tableau (range from 0 to 3).

e. Parameters: `tableau (list of lists), foundation (list), from_col (int)`
f. Returns: `nothing`
g. Display: nothing

`validate_move_within_tableau( tableau, from_col, to_col ) → bool`

a. The program will use this function to determine if a requested move to within the tableau is valid.
j. This function has three parameters: the data structure representing the tableau, an **`int`** indicating the index of the column whose bottom card should be moved, and an **`int`** indicating the column the card should be moved to. Check the game rules section to see when a move within the tableau is valid.
b. The function will return **`True,`** if the move is valid; and **`False`**, otherwise. In the latter case, it will also print an appropriate error message.
c. `from_col` and `to_col` are the indices of the columns in the tableau (range from 0 to 3).
d. The error messages should look like this:

```
tableau  = [[ A♥], [], [ 2♦,  3♥], [ 4♠,  5♦,  6♦]]
validate_move_within_tableau( tableau, 1, 0) returns False
Error, target column is not empty: 1
```

`validate_move_within_tableau( tableau, 1, 2)` returns `False`
`Error, target column is not empty: 1`

`validate_move_within_tableau( tableau, 3, 2)` returns `True`
tableau should not get modified.

`Tableau before: [[], [], [ 2♦,  3♥], [ 4♠,  5♦,  6♦]]`
`validate_move_within_tableau( tableau, 1, 0)` returns `False`
`Error, no card in column: 2`

e. Parameters: `tableau (list of lists), from_col (int), to_col (int)`
f. Returns: `bool`
g. Display: error messages

`move_within_tableau( tableau, from_col, to_col )` → `None`

a. The program will use this function to move a card from the tableau to the foundation.
b. This function has three parameters: the data structure representing the tableau, an **`int`** indicating the column whose bottom card should be moved, and an **`int`** indicating the column the card should be moved to.
c. If the move is valid, the function will update the tableau; otherwise, it will do nothing to it.
d. `from_col` and `to_col` are the indices of the columns in the tableau (range from 0 to 3).
e. Parameters: `tableau (list of lists), from_col (int), to_col (int)`
f. Returns: `nothing`
g. Display: nothing

`check_for_win( tableau, stock )` → `bool`

a. The program will use this function to check if the game has been won.
b. This function has two parameters: the data structure representing the stock and the data structure representing the tableau.
c. It returns **`True`**, if the stock is empty and the tableau contains only the four aces; and **`False`**, otherwise.

`display( stock, tableau, foundation )` → `None`

a. The program will use this function to display the current state of the game.
b. This function is already provided.

`get_option()→ list`

a. The program will use this function to prompt the user to enter an option and return a representation of the option designed to facilitate subsequent processing.
b. This function takes no parameters. It prompts the user for an option and checks that the input supplied by the user is of the form requested in the menu. If the input is not of the required form, the function prints an error message (error in option) and returns an empty list.
c. The program will recognize the following commands (upper or lower case):

`D`	Deal four cards from the stock to the tableau
`F x`	Move card from Tableau column **`x`** to the Foundation.
`T x y`	Move card from Tableau column **`x`** to empty Tableau column **`y`**.
`R`	Restart the game (after shuffling)
`H`	Display the menu of choices
`Q`	Quit

where **`x`** and **`y`** denote column numbers, and the columns are numbered from 1 to 4.

d. The function returns a list as follows:

- **`[]`**, if the input is not of the required form
- **`['D']`**, for deal
- **`['F', x]`**, where **`x`** is an **`int`**, for moving a card to the foundation
- **`['T', x, y]`**, where **`x`** and **`y`** are **`int`**'s, for moving a card within the tableau
- **`['R']`**, for restart
- **`['H']`**, for displaying the menu
- **`['Q']`**, for quit

`main():`

a. The program will have a main function.
b. The main will initialize the game, display the rules of the game, get an option and then loop until quitting. No error checking is done in main. All error checking for option is done in the `get_option` function. A message will be printed when restarting, quitting, and winning. See the `strings.txt` file for the detailed messages.
a. 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.
b. The program will detect, report, and recover from invalid commands. None of the data structures representing the stock, tableau, or foundation will be altered by an invalid command.

Assignment Notes

1. Before you begin to write any code, play with the provided link to the tutorial 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/: Under the "Solitaire" tab, click on "Select Game…" and "Aces Up".

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 requires aces to have a rank higher than any other card. Your program must implement "aces high" *without modifying the* **`cards`** *module***.**

3. Laboratory Exercise #10 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. Gradually replace the "stub" code (marked with comments) with your own code. (Delete the stub code.)

5. A `strings.txt` file is provided.

6. 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.

7. 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.

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

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

Sample Output

```
Aces High Card Game:
     Tableau columns are numbered 1,2,3,4.
     Only the card at the bottom of a Tableau column can be moved.
     A card can be moved to the Foundation only if a higher ranked card
        of the same suit is at the bottom of another Tableau column.
     To win, all cards except aces must be in the Foundation.

Input options:
    D: Deal to the Tableau (one card on each column).
    F x: Move card from Tableau column x to the Foundation.
    T x y: Move card from Tableau column x to empty Tableau column y.
    R: Restart the game (after shuffling)
    H: Display the menu of choices
    Q: Quit the game


stock      tableau     foundation
 XX      J♥  3♣  2♥  5♥

Input an option (DFTRHQ): f 4

stock      tableau     foundation
 XX      J♥  3♣  2♥          5♥

Input an option (DFTRHQ): f 3

stock      tableau     foundation
```

```
 XX      J♥  3♣              2♥

Input an option (DFTRHQ): f 1
Error, cannot move  J♥.

stock      tableau     foundation
 XX      J♥  3♣              2♥

Input an option (DFTRHQ): f 3
Error, empty column: 3

stock      tableau     foundation
 XX      J♥  3♣              2♥

Input an option (DFTRHQ): t 2 3

stock      tableau     foundation
 XX      J♥      3♣          2♥

Input an option (DFTRHQ): t 2 3
Error, target column is not empty: 3

stock      tableau     foundation
 XX      J♥      3♣          2♥

Input an option (DFTRHQ): d

stock      tableau     foundation
 XX      J♥  A♣  3♣  4♥      2♥
         K♠      A♦

Input an option (DFTRHQ): h

Input options:
    D: Deal to the Tableau (one card on each column).
    F x: Move card from Tableau column x to the Foundation.
    T x y: Move card from Tableau column x to empty Tableau column y.
    R: Restart the game (after shuffling)
    H: Display the menu of choices
    Q: Quit the game


stock      tableau     foundation
 XX      J♥  A♣  3♣  4♥      2♥
         K♠      A♦

Input an option (DFTRHQ): d

stock      tableau     foundation
 XX      J♥  A♣  3♣  4♥      2♥
         K♠  5♦  A♦  J♣
         6♥      3♠

Input an option (DFTRHQ): d

stock      tableau     foundation
 XX      J♥  A♣  3♣  4♥      2♥
         K♠  5♦  A♦  J♣
         6♥  8♥  3♠  8♦
         Q♠      5♠
```

```
Input an option (DFTRHQ): f 3

stock      tableau     foundation
 XX      J♥  A♣  3♣  4♥      5♠
         K♠  5♦  A♦  J♣
         6♥  8♥  3♠  8♦
         Q♠

Input an option (DFTRHQ): f 3

stock      tableau     foundation
 XX      J♥  A♣  3♣  4♥      3♠
         K♠  5♦  A♦  J♣
         6♥  8♥      8♦
         Q♠

Input an option (DFTRHQ): f 3
Error, cannot move  A♦.

stock      tableau     foundation
 XX      J♥  A♣  3♣  4♥      3♠
         K♠  5♦  A♦  J♣
         6♥  8♥      8♦
         Q♠

Input an option (DFTRHQ): f 4

stock      tableau     foundation
 XX      J♥  A♣  3♣  4♥      8♦
         K♠  5♦  A♦  J♣
         6♥  8♥
         Q♠

Input an option (DFTRHQ): r
=========== Restarting: new game ============

Aces High Card Game:
     Tableau columns are numbered 1,2,3,4.
     Only the card at the bottom of a Tableau column can be moved.
     A card can be moved to the Foundation only if a higher ranked card
        of the same suit is at the bottom of another Tableau column.
     To win, all cards except aces must be in the Foundation.

Input options:
    D: Deal to the Tableau (one card on each column).
    F x: Move card from Tableau column x to the Foundation.
    T x y: Move card from Tableau column x to empty Tableau column y.
    R: Restart the game (after shuffling)
    H: Display the menu of choices
    Q: Quit the game


stock      tableau     foundation
 XX      J♥  3♣  2♥  5♥

Input an option (DFTRHQ): f 2
Error, cannot move  3♣.

stock      tableau     foundation
```

```
 XX      J♥  3♣  2♥  5♥

Input an option (DFTRHQ): f 3

stock      tableau     foundation
 XX      J♥  3♣      5♥      2♥

Input an option (DFTRHQ): t 1 3

stock      tableau     foundation
 XX        3♣  J♥  5♥      2♥

Input an option (DFTRHQ): d

stock      tableau     foundation
 XX      K♠  3♣  J♥  5♥      2♥
            A♣  A♦  4♥

Input an option (DFTRHQ): x

Error in option: x

Input an option (DFTRHQ): z 1

Error in option: z 1

Input an option (DFTRHQ): d 2 3

Error in option: d 2 3

Input an option (DFTRHQ): q
You have chosen to quit.
```

Grading Rubric

Computer Project #10 Scoring Summary

General Requirements:

(5 pts) Coding Standard 1-9
(descriptive comments, function headers, mnemonic identifiers, format, etc...)

Implementation:

(6 pts) init_game function

(5 pts) deal_to_tableau function

(6 pts) get_option function

(8 pts) validate_move_to_foundation function

(4 pts) move_to_foundation function

(8 pts) validate_move_within_tableau function

(4 pts) move_within_tableau function

(5 pts) check_for_win function

(7 pts) Test 1

(7 pts) Test 2 (test that wins)

Note: hard coding an answer earns zero points for the whole project
-10 points for not using main()