HW7: CS 110X C 2013

Note: This final homework is a **partner homework** and must be completed by each partner pair. When you complete this assignment, you must not share your answers with any other student. Only one person from a partner pair needs to submit the assignment.

Please make sure that **as a team** you work together on these problems, but you also each individually understand the code for each of the associated programs.

This entire homework revolves around the game of Five-Card Draw Poker. For simplicity there will be no betting, but rather, the player that wins scores a point (I know, this takes all the psychology out of the game, but you can always go to Foxwoods if you'd like to play the real thing).

Q1 Create A Random Deck Of Cards **(7 points)**

To play poker you need a random deck of playing cards. As you may know a deck is composed of thirteen cards with values (Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen and King) drawn from four suits (Clubs, Diamonds, Hearts, Spades). The values are abbreviated as (A,2,3,4,5,6,7,8,9,10,J,Q,K) while the suits are abbreviated as (C,D,H,S).

Note that the template file provides you with two tuples, `cardValues` and `suitValues`, that you should take advantage of for this assignment.

Write a function `initializeDeck()` that returns a list of 52 string values in the default order. Each string is the concatenation of a value and suit. For example, "3H" represents the three of Hearts, "10C" represents the 10 of Clubs, and "JD" represents the Jack of Diamonds. The first 13 cards are Clubs, in order from Ace through King. The next 13 cards are Diamonds (in the same order), then 13 Hearts cards, and finally 13 Spades cards.

Sample Output

```
>>> initializeDeck()
['AC', '2C', '3C', '4C', … ]
```

Q2 Helper Function **(17 points)**

Write a Python function `randomDeck(seed)` which returns a shuffled deck. To do so, it shuffles an initial deck – as produced by `initializeDeck()` – by executing the following procedure 50 times: select two random cards in the deck and swap their locations. Note: Don't bother to check whether the two random cards are different locations.

Sample Output

```
>>> randomDeck(2013)
['8D', '4S', '4D', '4C', '2C', '6C', '10D', …
```

Hint: If you use the random seed provided, and swap 50 pairs of randomly selected cards, you should get the output above. Don't worry if you don't, however.

Q3	Helper Function **(11 points)**

Write a function `getValue(card)` that returns an integer value in the range 1—13 given a string representation of a card, as found in a deck computed in Q1. Return –1 if the card string is invalid.

Sample Output

```
>>> getValue('AD')
1
>>> getValue('JS')
11
```

Q4	Helper Function **(7 points)**

Write a function `getSuit(card)` that returns a string (either 'C', 'D', 'H' or 'S') given a string representation of a card as described in Q1. Return '' (empty string) if card string is invalid.

Sample Output

```
>>> getSuit('AD')
'D'
>>> getSuit('JS')
'S'
```

Q5	Helper Function **(9 points)**

Write a function `isFlush(hand)` that determines whether `hand`, a list containing five card representations, is a flush. This function returns `True` or `False`.

Sample Output

```
>>> isFlush(['3H','9H','5H','KH','AH'])
True
```

*Note: A **Flush** is a hand where all the cards belong to the same suit (above, for example, they are all Hearts cards).*

Q6	Helper Function **(12 points)**

Write a function `isStraight(hand)` that determines whether `hand`, a list containing five card representations, is a straight. This function returns `True` or `False`.

Sample Output

```
>>> isStraight(['3C','6D','5H','4H','7S'])
True
>>> isStraight(['JC','QD','2S','KH','AC'])
False
```

In a Straight, the values of the cards in the hand form a consecutive sequence, without any gaps. Thus the first example is a 3-4-5-6-7 straight.

Hint: Think about extracting the values of each card into a list, and then sorting that list.

*Note: In a **Straight**, an ACE can either be Low (A-2-3-4-5) or High (10-J-Q-K-A). You can't "wrap around" the Ace as the second example above shows.*

Q7 Statistics For Individual Hands **(17 points)**

In a 5-card hand of Poker there are specific hands that can be identified. These are:

Hand	Approx. Probability
Royal flush	0.0000015400
Straight flush (not including royal flush)	0.0000139000
Four of a kind	0.0002400000
Full house	0.0014400000
Flush (excluding royal flush and straight flush)	0.0019700000
Straight (excluding royal flush and straight flush)	0.0039200000
Three of a kind	0.0211000000
Two pair	0.0475000000
One pair	0.4230000000
No pair / High card	0.5010000000

I am asking you to compute the probability of three of these hands

- Straight Flush
- Straight
- Flush

To do so, try 100,000 trials where you create a random deck, using the trial number which ranges from 0 through 99,999 as the "seed" value for the random deck. Each random deck has ten 5-card hands (positions 0 – 4, 5 – 9, 10 – 14, …, 45 – 49 in the deck). This gives a total of 1,000,000 random poker hands.

Write a function `computeThreeProbabilities(numTrials)` that returns a tuple with three values (`numFlush, numStraight, numStraightFlush`), reflecting the count of each hand seen in the 10*numTrials of total random hands.

Note that when counting a straight flush, you must not also increment the counts for straight and flush (as the note in the above table suggests).

Q8 Approximating Probabilities **(18 points)**

Chevalier de Mere was a rich nobleman who gambled frequently. He posed two questions to Blaise Pascal in 1654 which essentially led to the foundation of modern probability theory.

(A) What is the chance of getting at least one 6 in four rolls of a single die?
(B) What is the chance of getting 12 at least once in 24 rolls of two dice?

He couldn't understand why he won more times gambling on (A) than he did with (B)! Write Python function `chevalierA(numTrials)` that returns probability for A. And write Python function `chevalierB(numTrials)` that returns probability for B.

Try for numTrials = 100,000 and see what your simulation gives you as a result.

Version: 2/20/2013 (2:00 PM) **Homework Due Tuesday 2/26/2013 5:00 PM**

How To Get Started On This Assignment

A template HW7.py file is provided to you.

You are responsible for properly documenting all functions as you have seen me do in class. The rubric will assign points for documentation, so pay attention!

Submit your HW7.py file using the web-based turnin system. As we have mentioned in class, only one of the team members needs to submit the assignment. But just make sure that something gets submitted!

Make sure that you don't write any additional code to invoke these functions, since that gets in the way of the TAs grading the assignments. Good Luck!