

GENERAL INSTRUCTIONS

This assignment consists of programming problems for which you must develop functions. Remember that it is unwise to code a solution before you understand the problem and have formulated an approach (a plan) which would solve it.

For all of the functions you must submit:

- A python function which is named and performs exactly as specified in the question and which will be submitted electronically in the file called `a4.py` as well as printed out and submitted with the paper copy of your assignment.
- An example of how you would test your coded solution (e.g., by running sample values through it) which will be submitted both electronically in the file `a4Test.py` and with the paper copy of your assignment. Testing is further discussed below.
- Be sure that you add **sufficient comments** to your functions and your tests to explain their purpose, the algorithm, the inputs and the outputs.
- Be sure to create your functions using the exact names specified in this assignment, and to pass the values in precisely the order they are shown. Failure to do this will cause the automarker (a program which will test your functions) to give you a zero for your function.

TESTING:

For this assignment we will test all of our functions in a separate python program `a4Test.py`. A starter function file `a4.py` and a sample function testing program file `a4Test.py` are available from the web.

When you test, try to think up some good examples. It is important to check that the output of a function is the value you expect. This can be accomplished simply by executing each function, and printing out the returned value.

SUBMITTING YOUR ASSIGNMENT:

Electronic submission:

Submit both `a2.py` and `a2Test.py` electronically using the submit command:

```
submit -c csc104h -a a4 a4.py a4Test.py
```

Submit printout:

You must submit a paper copy of your assignment containing the full development steps of your algorithms (where required), the code for all of your functions, and the code for your test cases. Staple the pages together with a completed copy of the cover sheet (provided on the website) attached to the front. Then put your paper copy in the drop box for this course (csc104) in the Bahen building room BA2220. You will lose many marks if your pages are not properly stapled together.

CONCEPTS AND HINTS to help you through this assignment:

1. Incremental development:

This is the same as A2 – but it is still important!!

Question: How do you eat an elephant sandwich?

Answer: One bite at a time

As you develop each function, test it immediately. It is easier to attack problems related to a single function than many problems related to many functions. It also helps you to identify mistakes you are making more quickly, before you can repeat them in subsequent functions.

2. Nesting lists:

We have learned to create and manipulate single lists. For this exercise, we will work with lists of lists.

We can nest any number of lists within a list. For example:

```
>>> M = [ [0,1, 2, 3], # A 4 x 4 matrix or table, as nested lists
          [4, 5, 6, 7],
          [8, 9, 10, 11] ]
>>> M
[[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]
```

You need to get creative and try this out in Wing. For example, try the following:

```
a = list()
a.append(list()) # create a new list nested in the list a
a.append(list()) # create another new list nested in the list a
a.append(list()) # create another new list nested in the list a

print len(a) # print the length of a (it should be 3)

# append items to individual lists within a
a[0].append("first")
a[0].append("second")
a[1].append("third")
a[2].append("fourth")
a[2].append("fifth")
a[2].append("sixth")

#verify length and contents of the lists
print a
print a[0]
print len(a[2])
print a[2]
print len(a[1])
print a[1]
```

3. Random selection:

This assignment will depend on randomly generated hands of cards. There is a module which can be used to randomly select from a list of items (such as cards). The module name is random. Below is an example of python code that uses the random function to select from a list.

```
import random          #the random module must be imported

a = ["red", "orange", "yellow", "green", "blue", "violet"]

# We use the random.choice() function with list a. One of
# the values from the list is randomly selected.
print "may you have a", random.choice(a), "day today!"
```

4. Cards and hands:

For the purposes of our program, we will represent our deck of cards as:

```
deck = [ \
'2c' , '3c' , '4c' , '5c' , '6c' , '7c' , '8c' , '9c' , '10c' , 'Jc' , 'Qc' , 'Kc' , 'Ac' , \
'2h' , '3h' , '4h' , '5h' , '6h' , '7h' , '8h' , '9h' , '10h' , 'Jh' , 'Qh' , 'Kh' , 'Ah' , \
'2s' , '3s' , '4s' , '5s' , '6s' , '7s' , '8s' , '9s' , '10s' , 'Js' , 'Qs' , 'Ks' , 'As' , \
'2d' , '3d' , '4d' , '5d' , '6d' , '7d' , '8d' , '9d' , '10d' , 'Jd' , 'Qd' , 'Kd' , 'Ad' ]

# note that we use the backslash \ to continue a line to the next line
```

Each quoted string in the list represents a card in the deck. The first character in each card is the rank, and the second character is the suit. An A represents an ace and can be considered low or high (i.e., a 1 or after a king), as required.

Each hand that we will deal for our poker game will consist of five cards. We will keep each hand in a separate list. For example:

```
hand1 = ['3d', '4s', '4h', '7c', 'Jc']
```

In situations where we deal more than one hand, we will nest all of the lists which contain a hand in a single list. For example:


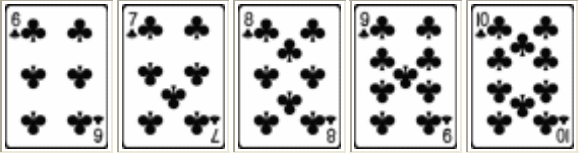

```
hands = [['3d', '4s', '4h', '7c', 'Jc'], ['Ac', 'Ah', 'As', 'Ad', 'Ks']]
```

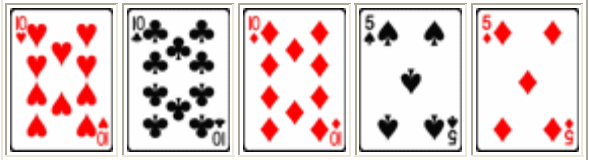
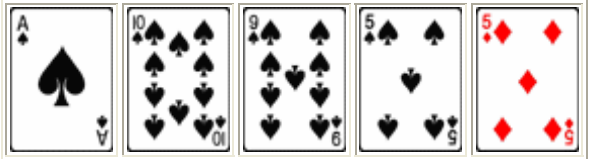
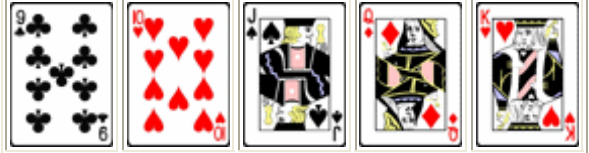
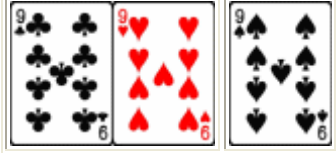
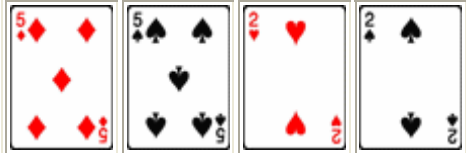
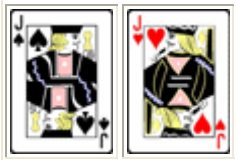
In this example we have nested two hands, for our game, we could nest up to ten hands.

5. Remember that you cannot deal a card more than once:

This assignment requires you to deal five cards to generate a Poker hand. You may use the random module to select a card. Remember that once a card has been selected and dealt to a hand, it cannot be used again until the next set of hands is dealt. Sometimes, however, the random routine selects the same item twice. When this occurs, you must make sure that your code handles the situation and prevents a duplicate card from being dealt. **Hint:** you may want to make use of parallel lists for this exercise or create a helper function that shuffles the deck (using the random function) before distributing the cards and then deals off the top of the shuffled deck.

6. Ranking hands: Hands are ranked as follows, from highest to lowest:

<p>Royal Flush This is the best poker hand you can have. Ex: Ten, Jack, Queen, King, and Ace of the same suit.</p>	
<p>Straight Flush Five cards, all of the same suit that are in sequence. If there are more than one straight flushes, the flush with the highest card wins. Ex: 6c, 7c, 8c, 9c, Tc</p>	
<p>Four-Of-A-Kind Four cards of equal rank. If there is more than one four-of-a-kind hand, the highest rank wins. Ex: Qc, Qh, Qd, Qs</p>	

<p>Full House Three cards of equal rank, and two different cards of the equal rank. (Three of a kind and a pair). If there is more than one full house, the full house with the highest ranking three-of-a-kind wins. Ex: Th, Tc, Td - 5s, 5d</p>	
<p>Flush Any five cards of the same suit. If there is more than one flush, the flush with the highest card wins. If the highest card ties, then look at the next highest card, etc. Ex: As, Ts, 9s, 5s, 2s</p>	
<p>Straight Five cards of mixed suits, in sequence. The highest straight wins, however there may be tying situations. Ex: 9c, Th, Js, Qd, Kh</p>	
<p>Three-Of-A-Kind Any three cards of equal rank. When more than one three-of-a-kind hands co-occur, the highest three-of-a-kind hand wins. Ex: 9c, 9h, 9s</p>	
<p>Two Pair Two cards of equal rank and a different two cards of equal rank. The highest pair wins. When there is a tie, look at the second pair. They may both tie. Ex: 5d, 5s - 2h, 2s</p>	
<p>One Pair Two cards of equal rank. The highest pair wins, but this hand may result in a tie when there is the same pair. Ex: Js, Jh</p>	
<p>High Card When there are no other winning card combinations, the player with the highest card wins. When there is a tie, then look at the next highest card ... and so on. There is a tie only if the players hold cards with the same ranks</p>	

If you are in doubt about the ranking of hands from the most winning to the least, be sure to discuss it on the course bulletin board. That way, everyone can get clarification at the same time.

Assignment description

Partly Poker wants to create a poker game website and requires you to develop some of the code for them. You are given a starter file that contains two sort functions. The first sorts a single hand by rank, the second, sorts the hand by suit and then rank within suit.

You must create the following functions in your program file a4.py. They must perform exactly as described. That is, if you are asked to return a list, return a list, if you are asked to return a nested list, return a nested list. Use the cards provided in the starter code. You may create additional lists for your own purposes, but the cards must be represented as in the deck description provided. Do NOT print out the returned values within the functions that you create. To test the functions, you may print out the values.

When you have finished each function, create one or more tests for it in the a4Test.py file. Get creative. For example, make sure that you test to ensure that no duplicates exist in the hands dealt, test ties, etc.

Here are the functions you must create:

Function	Specifications
dealHands(nbrHands)	Accept the number of hands to be dealt (nbrHands), and use the random function to deal that number of hands. Each hand must contain five cards. No card across all of the hands may be duplicated. For example, if one hand has the three of spades (3s), that hand cannot contain another three of spades, and no other hand that is dealt may also the three of spades. When dealing hands, you must distribute a single card to each hand, then a second card to each hand ... and so on, until five cards have been dealt to each hand. Return the hands dealt as lists nested within a single list, where each list is a hand. If there is only one hand, it must be nested as well.
royalFlush(hand)	Accept a hand (a list containing five cards). Return True if the hand is a royal flush. Otherwise, return False.
straightFlush(hand)	Accept a hand. Return True if the hand is a straight flush. Otherwise, return False.
fourOfAKind(hand)	Accept a hand. Return True if the hand is four-of-a-kind. Otherwise, return False.
fullHouse(hand)	Accept a hand. Return True if the hand is a full house. Otherwise, return False.
flush(hand)	Accept a hand. Return True if the hand is a flush. Otherwise, return False.
straight(hand)	Accept a hand. Return True if the hand is a straight. Otherwise, return False.
threeOfAKind(hand)	Accept a hand (a list containing five cards). Return True if the hand is three-of-a-kind. Otherwise, return False.
twoPair(hand)	Accept a hand. Return True if the hand is two pair. Otherwise, return False.
pair(hand)	Accept a hand. Return True if the hand is a pair. Otherwise, return False.

CSC104 – Assignment 4
Python

<code>winner(allHands)</code>	<p>Accept a nested list containing more than one hand. Compare the hand and return a nested list containing only the winning hand or the hands that tie for the win (if a tie occurs.)</p> <p>Hint: use the functions you have created to identify the rank of each hand in the winning scale. Then add logic, where required, to break ties. Be sure to follow the instructions for determining the winner that are described earlier in this assignment.</p>
-------------------------------	--