Compsci 101: Test 2 Practice

In writing code you do not need to worry about specifying the proper `import statements`. Do not worry about getting function or method names exactly right. Assume that all libraries and packages we have discussed are imported in any code you write.

PROBLEM 1 : (*Stun Chaos Tipi (24 points)*)

You'll be asked to write code that references the list `nuts` below. The Python code you write should work with any values stored in the list `nuts`. *You can write one line of code or many for each of the tasks below.*

```
nuts = ["cashew", "filbert", "chestnut", "coconut", "macadamia", "peanut", "pecan" \
        "peanut", "peanut", "macadamia", "cashew"]
```

Part A (4 points)

Write code to store in variable `unic` the number of different values stored in `nuts` that begin with the letter 'c' (in the example this value is 3: `"cashew"`, `"chestnut"`, `"coconut"`).

Part B (4 points)

Write code to store in `solo` the number of strings in `nuts` that occur exactly once, this would be four in the example above: `"filbert"`, `"chestnut"`, `"coconut"`, and `"pecan"` each occur once.

Part C (4 points)

Write code to store in `truenut` a list of the (unique) strings in `nuts` that end with `'nut'`. In the example above this is `[chestnut,coconut,peanut]` (order of words doesn't matter)

Part D (6 points)

Write code to store in `nuttiest` the string that occurs most often in `nuts`. This is `"peanut"` in the example above. Assume the string that occurs most often is unique, don't worry about breaking ties.

Part E (6 points)

Write code to store in `freqs` the unique strings in `nuts` sorted by number of times each string occurs with the least frequently occuring string first. For strings that occur the same number of times, break ties alphabetically. For example, for the list `nuts` above the value stored in `freqs` is `["chestnut", "coconut", "filbert", "pecan", "cashew", "macadamia", "peanut"]`.

PROBLEM 2 : (*Turn Turn Turn (12 points)*)

Consider the problem of determining if one string is a rotation of another. For example, "hello" has the following rotated versions: "lohel", "ohell", and "elloh"; but "elolh" is not a rotation because the last three letters are out of order.

For each of the three **correct** implementations given below, explain what it does and why it works as a solution to the problem. Note, for simplicity, you can assume the two strings being compared have the same length.

Part A

```
def isRotation (original, rotated):
    return original in 2*rotated
```

Part B

```
def isRotation (original, rotated):
    for i in range(len(original)):
        if rotated.startswith(original[i:]) and rotated.endswith(original[:i]):
            return True
    return False
```

Part C

```
def isRotation (original, rotated):
    return sorted([ original[k:] + original[:k] for k in range(len(original)) ]) ==
           sorted([ rotated[k:] + rotated[:k] for k in range(len(rotated)) ])
```

PROBLEM 3 : (*Big Ugly Gigantic Spiders (15 points)*)

The *highCard* function returns the maximum number of points a cheater can win given two lists of integers representing two hands of cards to be played against each other in rounds. A player earns a point by playing the higher card for that round. The cheater knows exactly what cards his opponent has and in what order; the cheater also can order his cards in any way he wants.

This table illustrates what the function is *supposed* to return:

ID	Call	Expected Value
#1	`highCard([2, 3, 4, 7], [3, 4, 3, 8])`	2
#2	`highCard([3, 3, 2, 2, 5, 3], [3, 4, 2, 9, 8, 2])`	3
#3	`highCard([2, 4, 6, 8], [3, 5, 7, 9])`	3

Consider the following solutions that are **not** correct. Each fails **at least one** of the examples shown above. Complete the table below for each solution, saying whether the given solution passes or fails the referenced test case and, if it fails, what the actual return value is. After the table explain, in general terms, what is wrong with the solution.

```
def highCard (mine, friend):
    points = 0
    for val in reversed(sorted(mine)):
        beatables = [x for x in friend if x < val]
        if len(beatables) > 0:
            points += 1
    return points
```

ID	Pass/Fail	Actual Value
#1		
#2		
#3		

Problem:

```
def highCard (mine, friend):
    for val in reversed(sorted(mine)):
        beatables = [x for x in friend if x < val]
        if len(beatables) > 0:
            friend.remove(beatables[0])
    return len(mine) - len(friend)
```

ID	Pass/Fail	Actual Value
#1		
#2		
#3		

Problem:

```
def highCard (mine, friend):
    points = 0
    for m in reversed(sorted(mine)):
        for f in reversed(sorted(friend)):
            if m > f:
                friend.remove(f)
                points += 1
    return points
```

ID	Pass/Fail	Actual Value
#1		
#2		
#3		

Problem:

PROBLEM 4 : (*Four star rating (20 points)*)

These questions ask you about the code from lab 7 regarding data about movies that is given as a handout with the exam. The first two questions ask you to describe how that code works, the last two ask you to write new code using the dictionary `movies`. In writing code for these questions, you may use any of the variables or functions available in the code.

Part A (4 points)

In the dictionary `movies`, the values are a list of strings, with some having length 9 and some having length 11. Explain the difference, what does it mean when there are 11 elements of movie data?

Also, for those movies that have 11 elements, what value must the final string represent?

Part B (4 points)

The function `processData` takes as its second parameter, `result`, a dictionary. Explain the purpose of this parameter given its two different values within the code:

```
movies = processData(rated, {})
```

```
movies = processData(rated, movies)
```

Part C (5 points)

Write a function that returns a list of tuples of an int and a string, (total money grossed, name of director), for the directors of the top grossing movies. The int, total money grossed, should be a sum of the gross profits of all movies directed by that director. The returned list should be in sorted order, by number from highest to lowest, and should be limited to only the top N directors as determined by the parameter `count`.

A partial output of the list returned might look like this:

```
[(3059836183, 'Steven Spielberg'), (1626418480, 'George Lucas'), ..., (623279547, 'Joss Whedon')]
```

Write the function `mostProfitableDirectors` below.

```
def mostProfitableDirectors (movies, count):
    """
    returns a list of tuples, (int money earned, string director name),
      of length count, the directors whose movies made the most money
    """
```

Part D (7 points)

Write a function that returns a list of tuples of an int and a string, (number of movies, decade), for the movies in the given dictionary, in sorted order by number from highest to lowest.

A partial output of the list returned might look like this:

```
[(157, '2000s'), (75, '1990s'), (72, '2010s'), (42, '1980s'), (31, '1950s'), ..., (4, '1920s')]
```

Note, a nice trick you can use to determine the decade is to ignore the last digit of the year in which the movie was produced:

```
decade = year[:3]
```

and then add the `'0s'` back in for the tuple to make it easier to read:

```
(count, decade + '0s')
```

Write the function `mostMoviesPerDecade` below.

```
def mostMoviesPerDecade (movies):
    """
    returns a list of tuples, (number of movies, decade name)
     for the given movies
    """
```