Homework #5 - Debugging/Test Cases

In this homework, you will be debugging several functions that have been provided for you. Your goal is to get them all to work correctly and pass the provided test cases. You may NOT adjust the test cases in any way. Here are descriptions of all of the functions:

getData(file)

Reads in a file and returns a list of dictionary objects where the keys are from the first row in the data and the values are each of the other rows

mySort(data, col)

Takes in a list of dictionaries and a column (ie. key) to sort on (alphabetically from a to z). Returns the first item in the sorted list as a string in the following format: "*firstName lastName*"

classSizes(data)

Takes in a list of dictionaries and returns a list of tuples sorted by the number of students in that class in descending order

Ex. return value: [('Senior', 26), ('Junior', 25), ('Freshman', 21), ('Sophomore', 18)]

findMonth(a)

Takes in a list of dictionaries and returns the most common birth month (ie. the month (1-12) that had the most births in the data)

calculateAgeFromDOB(dob)

Takes in the date of birth (month/day/year) and returns the age (in years) from that date of birth.

findAge(a)

Takes in a list of dictionaries and returns the average age of the students (round that age to the nearest) integer. You will need to work with the DOB and the current date to find the current age in years.

Add 2 Test Cases:

Add two test cases (assert statements) to the test method for *test_calculate_age_from_DOB*.

Grading Rubric (60 points):

Note that if you use hardcoding (specify expected values directly) in any of the functions that you are editing to get them to pass the test cases, you will NOT receive credit for passing that test case.

6 points for passing test_calculate_age_from_DOB
6 points for test_calculate_average_age
6 points for test_data_read_in_correctly

6 points for test_mySort_email
6 points for test_mySort_first_name
6 points for test_mySort_last_name

10 points for test_each_grade_ordered_by_size
10 points for test_most_common_birth_month

4 points for additional test cases for *test_calculate_age_from_DOB* (2 points per test case)

Sample Output:

```
test_calculate_age_from_DOB (__main__.TestHomework5) ... ok
test_calculate_average_age (__main__.TestHomework5) ... ok
test_data_read_in_correctly (__main__.TestHomework5) ... ok
test_each_grade_ordered_by_size (__main__.TestHomework5) ... ok
test_most_common_birth_month (__main__.TestHomework5) ... ok
test_mySort_email (__main__.TestHomework5) ... ok
test_mySort_first_name (__main__.TestHomework5) ... ok
test_mySort_last_name (__main__.TestHomework5) ... ok

----------------------------------------------------------------------
Ran 8 tests in 0.009s

OK
```

Extra Credit

Part 1 (3 points)

Implement a function called *myNthSort*, which takes in three parameters: data, col, and n

This will function very similar to *mySort*, except it will return a list of the first n items in the sorted list in the following format:

ex.) if n = 2

['firstName1 lastName1', 'firstName2 lastName2']

Part 2 (3 points)

Create a non-trivial test case with at least two assert statements for the above *myNthSort* function.