Important

1. Due Date: **Monday, March 13th**
2. This homework is graded out of 100 points.
3. This is an Individual Assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.
4. For Help:
    - TA Helpdesk (Schedule posted on class website.)
    - Email TA's or use Piazza Forums Notes:
    - How to Think Like a Computer Scientists [http://openbookproject.net/thinkcs/python/english3e/]
    - CS 1301 Python Debugging Guide [http://www.cc.gatech.edu/classes/AY2016/cs1301_spring/CS-1301-Debugging-Guide/index.html]
5. Don't forget to include the required collaboration statement (outlined on the syllabus). Failing to include the Collaboration Statement will result in no credit.
6. Do not wait until the last minute to do this assignment in case you run into problems.
7. Comment or delete all your function calls. When your code is run, all it should do is build without any errors. Having function calls or extraneous code outside the scope of functions will result in a lower grade. (import statements and global variables are okay to be outside the scope of a function)
8. **Read the entire specifications document before starting this assignment.**
9. **IF YOUR CODE CANNOT RUN BECAUSE OF AN ERROR, IT IS A 0%.**

Introduction

The goal of this homework is for you to showcase your knowledge about how to properly use dictionaries. For this assignment you will need to implement 5 functions. Refer to the rubric to see how points will be rewarded for each function. You have been given HW7.py to fill out with instructions in the docstrings. However, below you will find more detailed information to complete your assignment. Read it thoroughly before you begin. You have until **Monday, March 13th** to complete this assignment. Don't forget to include your name and your collaboration statement. Re-download your submission from T-Square after you submit it to make sure that your code runs successfully.

Function name: `charCount`
Parameters: a string
Returns: a dictionary

Description: Make a function called `charCount` that takes in a string as a parameter and returns a dictionary. The dictionary should have each character in the string as a key and the corresponding value should be the amount of occurrences of that character.

Test cases:

```
>>> charCount("hello")
{'o': 1, 'l': 2, 'e': 1, 'h': 1}
>>> charCount("Quavo Ratatouille")
{'Q': 1, 'u': 2, ' ': 1, 't': 2, 'v': 1, 'e': 1, 'R': 1, 'i': 1, 'l': 2, 'a': 3, 'o': 2}
>>> charCount("Alabama")
{'A': 1, 'm': 1, 'b': 1, 'l': 1, 'a': 3}
```

Function name: `shopping`
Parameters: a dictionary {key is the name of an item (str): value is a tuple containing the item's cost per unit (float) and how many of that item you want to buy (int)), a list of items that are on sale (str)
Returns: a dictionary {key is the <str> name of an item: value is <float> how much was actually paid for it}

Description: Write a function that takes in a dictionary in which every key is the name of an item, and every value is a tuple with how much the item costs per unit, and how many of the item you want to buy, as well as a list of things that qualify for the store's 20% off sale. The function should return a dictionary in which every key is the name of an item you bought, and the value is the total amount you paid for that item (i.e. unit price * quantity * .8 *if* the item is on sale). The dictionary should also have a key-value pair where the key is "total" and the value is the grand total for all of the things you bought.

Notes:

- All prices in the returned dictionary should be rounded to 2 decimal points.
- You can assume that all the items in the list of items on sale will be lowercase. However, not all of the items in the original dictionary will be lowercase.
- You can assume that "total" will never be a key in the original dictionary.

Test cases:

```
>>> shopList = shopping({"apple" : (.5,10), "laptop" : (700,1), "chicken nugget" : (.25,8), "coffee" : (3,3), "APPLE" : (.5,10), "umbrella" : (9,1), "pillow" : (20,2), "cereal" : (3,1)}, ["apple", "pillow"])
>>> print(shopList)
>>> {'apple': 4.0, 'laptop': 700, 'chicken nugget': 2.0, 'coffee': 9, 'APPLE': 4.0, 'umbrella': 9, 'pillow': 32.0, 'cereal': 3, 'total': 763.0}
```

Function name: `dictReplaceStr`
Parameters: a string, a dictionary
Returns: a string

Description: Write a function that takes in a string and a dictionary. The key to dictionary will be a string and the value is another string. In the string, replace every word that is a key in the dictionary with the value that the key maps to and return that new string.

Test cases:

```
>>> clean = {"brb": "be right back", "asap": "as soon as possible", "ily": "I
love you", "lol":"laugh out loud"}

>>> message = "Hey bring me that asap"
>>> test1 = dictReplaceStr(message,clean)
>>> print(test1)
Hey bring me that as soon as possible

>>> message2 = "lol that was really funny, ily"
>>> test2 = dictReplaceStr(message2,clean)
>>> print(test2)
laugh out loud that was really funny, I love you

>>> lingo = {"like":"about", "Young":"Yung", "like":"about", "you":"we",
"Hey":"Aye"}

>>> print( dictReplaceStr("Yo you like this?", lingo) )
Yo we about this?

>>> print( dictReplaceStr("Hey Young Margs I hear you dibble dabble with the
Snapple", lingo) )
Aye Yung Margs I hear we dibble dabble with the Snapple

>>> print( dictReplaceStr("Hey YOung Margs I hear you dibble dabble with the
Snapple", lingo) )
Aye YOung Margs I hear we dibble dabble with the Snapple
```

Function name: `groupAge`
Parameters: a dictionary {keys are strings: values are dictionaries with member names as keys and their ages as values}
Returns: a dictionary {keys are strings of the group: values are integers with group artists' ages}

Description: Write a function that takes in a dictionary and returns a dictionary. The function will take in a dictionary where the key will be a string that represents the name of some group and the value will be another dictionary that has the members of the group as the key and their age as the value. The function should take this dictionary and make a new dictionary where the keys will be strings of the group name and the values will be the total of all band member's ages.

```
>>> aDict = {'AWA': {'Yung Coco Butter': 20, 'Yung Erica': 21}, 'OutKast':
{'Big Boi': 42, 'Andre 3000': 3000}, 'Migos': {'Quavo': 25, 'Offset': 23,
'Takeoff': 26}, 'Michael Jackson': {'Michael': 50}}
>>> test1 = groupAge(aDict)
>>> print(test1)
{'AWA': 41, 'OutKast': 3042, 'Migos': 74, 'Michael Jackson': 50}
```

Function name: `statistics`
Parameters: a list of numbers (ints or floats)
Returns: a dictionary {keys are strings: values are lists or numbers}

Description: Write a function that takes in a list of numbers and returns a dictionary with some basic statistical measures (see table below) about the data in the list.
Notes:

- You can **NOT** assume that the given list will be in numerical order.
- The `dict.values()` function, which returns a list of the values in a dictionary, might be useful for this function.
- You may find it helpful to import the math module for this function.
- We encourage you to write your own helper functions if you will like to simplify your code.
- You can **NOT** use the `statistics` module. Doing so will result in 0% for this question.
- Make sure every number in the returned dictionary is rounded **to 3 decimal places**.

Key	Value
"min"	The smallest number found in the list
"max"	The largest number found in the list
"sum"	The total of all the numbers in the list
"mean"	The average value found in the list, rounded to 3 decimal places
"median"	The middle value found in the list, rounded to 3 decimal places; if there is an even number of elements in the list, this value is the average of the two middle elements

"mode"	A list of the values that appear the most times in the list. If there is more than one mode, make sure the list is sorted numerically
"var"	The variance of the data in the list. To calculate this, follow these steps: 1. Find the mean 2. For each number in the list, find the difference between the number and the mean, and square the result 3. Add all these squared differences together. 4. Find the average of this total sum; this is the variance. Round it to 3 decimal places. For more information, visit this site.
"sdev"	The standard deviation of the data in the list. To calculate this, just take the square root of the variance, calculated above. Round it to 3 decimal places. For more information, visit this site.

Test cases:

```
>>> statDict = statistics([33, 4, 5, 777, 6, 5, 8254, 6, 8, 1, 3.547, 4.67,
2,89, 0, 44, 7, 5, 2, 5, 7, 888, 8, 7, 7, 654, 62, 4.333667, 2.79547])
>>> print(statDict)
{'min': 0, 'sdev': 1507.063, 'sum': 10901.346, 'mean': 375.908, 'max': 8254,
'median': 6, 'var': 2271240.291, 'mode': [5, 7]}
```

Grading Rubric

```
- charCount:           20 points
- shopping:            20 points
- dictReplaceStr:      20 points
- groupAge:            20 points
- statistics:          20 points

_____________________________________________
- Total                100/100 points
```

Provided

The following file(s) have been provided to you. There are several, but you will only edit one of them:

1. HW7.py

This is the file you will edit and implement. All instructions for what the methods should do are in the docstrings.

2. HW7_test.py

We have provided you with a python file called `HW7_test.py`. In this file we have created a series of tests for your usage. We understand you probably have never been exposed to testing code so you are not expected to do anything to this file or even use this file if you don't want to. However, we encourage you to use it as it will be highly beneficial in testing your code. Feel free to add your own tests to the file to cover any additional cases you would like to test.

If you do desire to test your code, all you have to do is have the `HW7.py` and the `HW7_test.py` files in the same directory. Open and run `HW7_test.py`. After running the test, you should see the results. Check the results and start debugging if needed. If you pass all the tests you should see something like this as your output:

Disclaimer: **The tests found in `HW7_test.py` are not intended to be an exhaustive list of all test cases and does not guarantee any type of grade. Write your own tests if you wish to ensure you cover all edge cases.**

Read more about unittest here: [https://docs.python.org/3/library/unittest.html]

Deliverables

You must submit all of the following file(s). Please make sure the filename matches the filename(s) below. Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder and run them.

1. HW7.py

If this file does not run (if it encounters an error while trying to run), you will get no credit on the assignment.