Important

1. Due Date: **Thursday, February 23rd**
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 tuples and modules. For this assignment you will need to implement 6 functions. The functions you are asked to code will use tuples and modules. Refer to the rubric to see how points will be rewarded for each function. You have been given `HW5.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 **Thursday, February 23rd** 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: **`complex_calculator`**
Parameters: a string, a list
Return value: An int, a float, or None

Description: Write a function that takes two arguments. The first argument will be a name (string). The second argument will be a list of numbers. The name will decide what function from the math module will be used. Refer to the table below to decide which name will correspond to which math function. If the name passed into the function is not one in the table, return None. The list of numbers contains the arguments to be used for the math function. However, some of these math functions take only 1 parameter and others take 2. Assume that if the math function needs one parameter, the list will be of length one, and if the math function needs two parameters the list will be of length two.

Name	**Function**
"ceil"	`math.ceil`
"fabs"	`math.fabs`
"gcd"	`math.gcd`
"pow"	`math.pow`
"hypot"	`math.hypot`

Notes:

- Assume that if a math function is called, the numbers in the list will work for that math function. This is not a case sensitive function (i.e. "ceil" is the same as "CEIL"), you must account for this.
- Look in the python library to see which functions take 1 parameter and which take 2. You may find that reading https://docs.python.org/3/library/math.html will be helpful.

Test Cases:

```
>>> test1 = complex_calculator("pow", [2, 4])
>>> print(test1)
16.0

>>> test2 = complex_calculator("code", [1, 2])
>>> print(test2)
None

>>> test3 = complex_calculator("gcd", [24, 8])
>>> print(test3)
8
```

Function name: **extract_information**
Parameters: A string
Return value: A tuple of length 3

Description: Write a function that takes in a string containing information about a person. You may assume that the string will always be formatted in this way: "NAME: AGE, SCHOOL". Return a tuple in the format (NAME (string), AGE (int), SCHOOL (string)).

Test Cases:

```
>>> test1 = extract_information("Buzz: 22, Georgia Tech")
>>> print(test1)
('Buzz', 22, 'Georgia Tech')

>>> test2 = extract_information("Dawg: 0, University of georgia")
>>> print(test2)
('Dawg', 0, 'University of georgia')

>>> test3 = extract_information("George: 102, University of Awesome")
>>> print(test3)
('George', 102, 'University of Awesome')
```

Function name: **create_addressbook**
Parameters: A list of strings
Return value: A list of tuples

Description: Write a function that takes in a list of strings. The list can be empty, in which case return an empty list. Each string contains information about a person. You may assume that the string will always be formatted in this way: "NAME: AGE, SCHOOL". Call the `extract_information` function to convert each string to a tuple. Add the tuples into a list. However, the tuples in the new list should be formatted (SCHOOL (string), NAME (string), AGE (int)). If there are two people with the same name, only the most recent information (the information last in the list) should be added. Return the list.

Notes:

- The returned list should be in the right order. Check test cases for situations where there are strings with the same name.
- Hint: If you have a function that returns a tuple of length three another way you can call the function and store the returned values is:
    - `a, b, c = myfunc()`

Test Cases:

```
>>> addressBook1 = create_addressbook(["Elena: 21, Georgia Tech",
"David: 20, UGA", "Eleven: 13, Hawkins Middle School", "David: 21,
Georgia Tech", "Claire: 18, Emory"])
```

```
>>> print(addressBook1)
[('Georgia Tech', 'Elena', 21), ('Hawkins Middle School', 'Eleven',
13), ('Georgia Tech', 'David', 21), ('Emory', 'Claire', 18)]

>>> addressBook2 = create_addressbook(["Adrienne: 18, High School",
"Niko: 12, Middle School", "Adrienne: 19, College"])
>>> print(addressBook2)
[('Middle School', 'Niko', 12), ('College', 'Adrienne', 19)]

>>> addressBook3 = create_addressbook(["George: 102, University of
Awesome", "Richard: 19, Alabama", "Mickey: 8, Tulane Elementary
School", "Nick: 15, Bellmont High School", "Carry: 19, Clemson",
"Richard: 22, MIT"])
>>> print(addressBook3)
[('University of Awesome', 'George', 102), ('Tulane Elementary
School', 'Mickey', 8), ('Bellmont High School', 'Nick', 15),
('Clemson', 'Carry', 19), ('MIT', 'Richard', 22)]
```

Function name: **get_averages**
Parameters: A list of tuples
Return value: A list of tuples

Description: Write a function that takes a list of tuples. The tuples in the list contains integers and floats. The list and tuples could be of any length (it can be empty). For each tuple, find the average of all the numbers and multiply each entry of the tuple by that average (each entry should be at max 2 decimal places). Return the new list of tuples.

Notes:

- Assume all numbers in the tuples are positive. In the returned list, all numbers in the tuples should be floats.

Test Cases:

```
>>> test1 = get_averages([(1,2,3), (4,5,9)])
>>> print(test1)
[(2.0, 4.0, 6.0), (24.0, 30.0, 54.0)]

>>> test2 = get_averages([(), (2.5, 7, 11), (6.5, 1.25, 8)])
>>> print(test2)
[(), (17.08, 47.83, 75.17), (34.12, 6.56, 42.0)]

>>> test3 = get_averages([(2,8), (1.0,7,9,3), (2.5, 4.1, 8.4)])
>>> print(test3)
[(10.0, 40.0), (5.0, 35.0, 45.0, 15.0), (12.5, 20.5, 42.0)]
```

Function name: **party_time**
Parameters: A list of tuples, an int
Return value: A tuple

Description: Write a function that takes in a list of tuples and an integer representing the max amount of people allowed in the party (size of the returned tuple). Each tuple with be comprised of a name of a person (string) and the priority status of that person (int) (status will be between 1 and 10, inclusively). You can assume a person with a priority status of 1 will be allowed in the party before someone of the priority status of 5. If more than one person has the same priority status, then you should give priority to the person who appears first in the list. When crafting the returned tuple, make sure to place the people in the tuple in order of priority. Return the tuple of guest names.

Notes:

- The returned tuple size can be less than the max.

Test Cases:

```
>>> party1 = party_time([('David', 2), ('Karoline K.', 7), ('Ryan',
9), ('Amanda', 4), ('Elena', 2), ('Will', 1)], 2)
>>> print(party1)
('Will', 'David')

>>> party2 = party_time([('Carlos', 1), ('Gretchen', 8), ('Abby', 6),
('Sam', 4)], 5)
>>> print(party2)
('Carlos', 'Sam', 'Abby', 'Gretchen')

>>> party3 = party_time([('Matt', 5), ('Ellen', 1), ('Chris', 10),
('Hannah', 4), ('Dom', 2), ('Ryan', 9), ('Kendall', 3)], 4)
>>> print(party3)
('Ellen', 'Dom', 'Kendall', 'Hannah')
```

Function name: **ocd**
Parameters: A list of 3 tuples
Return value: A list of 3 tuples

Description: Write a function that takes in a list of 3 tuples. The tuples could be of any length (however, assume it will not be empty). The tuples will contain ints and floats. Make a new list of three tuples. The tuples should be the same lengths as the ones passed in. However, the content should be sorted in ascending order. In other words, sort the contents of the list while keeping the lengths of all the tuples the same. Return the list.

Hint:

- Place all entries from the tuples into one list and then use the .sort() function

Test Cases:

```
>>> test1 = ocd([(4, 2, 6), (6, 4, 2, 4, 2), (1,)])
>>> print(test1)
[(1, 2, 2), (2, 4, 4, 4, 6), (6,)]

>>> test2 = ocd([(-1, 7, 3, 8), (2, 11, 73, 7), (1,)])
>>> print(test2)
[(-1, 1, 2, 3), (7, 7, 8, 11), (73,)]

>>> test3 = ocd([(1.4, 2, 1.2, 43.0, 0.0)])
>>> print(test3)
[(0.0, 1.2, 1.4, 2, 43.0)]
```

Grading Rubric

```
- complex_calculator: 10 points

- extract_information: 10 points

- create_addressbook: 10 points

- get_averages: 20 points

- party_time: 25 points

- ocd: 25 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. HW5.py
   This is the file you will edit and implement. All instructions for what the methods should do are in the docstrings.

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. HW5.py
   If this file does not run (if it encounters an error while trying to run), you will get no credit on the assignment.