Important

1. Due Date: **Thursday, September 14th**
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 implement the different types of loops you have learned. The functions you are asked to code will all use some looping technique and certain loops may be better fit for a certain purpose. Although some of the functions can be completed without the use of a loop, you are being graded on your ability to implement different looping techniques. Refer to the rubric to see how points will be rewarded for each function. You have been given HW3.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, September 14th to complete this assignment.

Function name (1): **`rearrange_vowels`**
Parameters: **`string to be iterated through (str)`**
Return value: **`string starting with all of the vowels of the original string, and ending with all of the consonants of the original string (str)`**

Description:
Write a function that takes in a string and separates all of the vowels from the consonants, and creates a new string made from the separated portions. The ordering of the vowels and consonants should be the same order that they are present in the string. Consider only "a","e","i","o", and "u" as vowels. "Y" should be considered a consonant. Capitalization should also be preserved. Any spaces present in the original string should be ignored and not added to the final output. You must use a for loop in this function to receive full credit.
Test Cases:

```
>>> rearrange_vowels("hello") —›  "eohll"
>>> rearrange_vowels("Oklahoma") —› "Oaoaklhm"
>>> rearrange_vowels ("Computer Science") —› "oueieeCmptrScnc"
```

Hint: The `lower()` string function might help in this function. Also look at Python's "in" operator when dealing with strings.

Function name (2): **`censor`**
Parameters: **`string to censor (str)`**
Return value: **`censored string (str)`**

Description:
You are working at a children's television network, and your job is to censor any words that could potentially be inappropriate. In order to do this, you have to replace certain letters with symbols. You must use a for loop in this function to receive full credit. The rules for censorship or as follows:

1. "a" or "A" → "!"
2. "u" or "U" → "*"
3. "i" or "I" → "@"
4. "e" or "E" → "#"

Test Cases:

```
>>> censor("What the heck!") —› "Wh!t th# h#ck"
>>> censor("You smell bad") —› "Yo* sm#ll b!d"
>>> censor("I love CS!") —› "@ lov# CS!"
```

Function name (3): **`odd_and_even_multiples`**
Parameters: **`non-zero lower bound of the range and number used to calculate multiples (int), upper-bound of a range (int)`**
Return value: **`string displaying the number of odds and evens (str)`** `(follow the format below exactly, not having the exact format will result in not receiving full credit for this function)`

Description:
Your function should iterate through a range from the lower bound of the range to the given upper-bound. Count the number of odd and even numbers that are multiples of the first parameter. Ensure that the upper bound is actually included in the range for this function. Be sure to return the string in the exact format below.

Test Cases:
```
      >>> odd_and_even_multiples(2, 10)—›
         "5 even multiple(s) and 0 odd multiple(s) from 2-10"

      >>> odd_and_even_multiples(3,14)—›
         "2 even multiple(s) and 2 odd multiple(s) from 3-14"

      >>> odd_and_even_multiples(1,10)—›
         "5 even multiple(s) and 5 odd multiple(s) from 1-10"

      >>> odd_and_even_multiples(-4,8)—›
         "4 even multiple(s) and 0 odd multiple(s) from -4-8"
```

Hint: Use the modulo operator to check if a number is odd or even.

Function name (4): **`most_common_character`**
Parameters: **`a word or sentence (str), a string containing characters that will be counted (str)`**
Return value: **`a string representing which character of the second parameter appears most frequently in the phrase string (str).`**

Description:
Write a function that receives two strings as parameters. The first parameter is a sentence, while the second string will contain any combination of characters (letters, numbers, space, special characters, etc.). Your code should find which of the given characters in the second parameter appears most frequently in the sentence string. Case should not be ignored, so for example, “A” and “a” should be considered two different characters. If no characters appear, an empty string should be returned. If there is a tie between multiple letters, return the character that appears first in the second parameter (random characters).

Test Cases:
```
      >>> most_common_character("This is a sentence", "!aTsn") —› "s"
      >>> most_common_character("Mississippi", "lmnop") —› "p"
      >>> most_common_character("Taylor Swift is bad", "mxz") —› ""
      >>> most_common_character("aaa bbb ccc dddddd", "cba") —› "c"
```

Function name (5): **`sum_multiples`**
Parameters: **`non-zero lower bound of the range and number used to calculate multiples (int), upper-bound of a range (int)`**
Return value: **`sum of the multiples found in the given range (int)`**

Description:
Your function should iterate through a range from the lower bound of the range to the given upper-bound, and, using the first parameter (which is also used for the lower bound), calculate the sum of any multiples found in the range. Ensure that the upper bound is actually included in the range for this function.

Test Cases:

```
>>> sum_multiples(2,6) —›  12     // 2 + 4 + 6
>>> sum_multiples(1,6) —›  21     // 1 + 2 + 3 + 4 + 5 + 6
>>> sum_multiples(5,18) —› 30     // 5 + 10 + 15
>>> sum_multiples(-3,6) —› 6      // -3 + 0 + 3 + 6
```

Function name (6): **`remove_vowels`**
Parameters: **`a word (str)`**
Return value: **`original word without vowels (str)`**

Description:
Write a function that takes a string as a parameter, removes the vowels from the string (upper and lowercase). It should return one string, consisting of the same letters of the original string, but without the vowels. Consider only “a”,”e”,”i”,”o”, and “u” as vowels. “Y” should be considered a consonant. **Must iterate through the string.**

Test Cases:

```
>>> remove_vowels("applE") —›  "ppl"
>>> remove_vowels("mississippi") —›  "mssspp"
>>> remove_vowels("logArithm") —›  "lgrthm"
```

Hint: String concatenation might help for this function. Remember that strings are immutable!

Function name (7): **`guess_dumplings`**
Parameters: **`number of dumplings ate (int)`**
Return value: **`number of guesses the user took (int)`**
Description:
Write a function that takes an integer value of the number of dumplings you ate, and asks the user to try to guess this number. When the user guesses the correct value, print a congratulatory statement and tell them how many guesses it took. If the user inputs “quit” (exactly the string quit, don’t worry about edge cases like ‘QUIT’ or ‘Quit’), your code should end, and print the correct answer. The integer returned if the user quits should be -1. If the user has not guessed the correct answer within the 5th try, print that they’ve lost the game, and return 0. **You must use a while-loop.**

Test Cases:

```
>>> a = guess_dumplings(5)
>>> “What is your guess?” —›  2
>>> “Wrong answer, try again” —›  4
>>> “Wrong answer, try again” —›  5
>>> “Correct! It took you 3 tries.”
_____________________________________
>>> print(a)
>>> 3

>>> a = guess_dumplings(5)
>>> “What is your guess?” —›  2
>>> “Wrong answer, try again” —›  4
>>> “Wrong answer, try again” —›  quit
>>> “The correct answer was 5.”
_____________________________________
>>> print(a)
>>> -1

>>> a = guess_dumplings(5)
>>> “What is your guess?” —›  1
>>> “Wrong answer, try again” —›  2
>>> “Wrong answer, try again” —›  3
>>> “Wrong answer, try again” —›  4
>>> “Wrong answer, try again” —›  6
>>> “You lose. The correct answer was 5.”
_____________________________________
>>> print(a)
>>> 0
```

Function name (8): **`tie`**
Parameters: **`a number 2-9 specifying half the max width of the tie (int)`**
Return value: **`N/A`**
Description:
Write a function that takes in a number specifying half the width of the tie as a parameter and prints a tie of those specifications. Make sure the tie prints in the correct format as shown in the example below. **Do not hardcode this. You must use a for-loop (or several).**

Here is an example of how the function should work:

```
>>> tie(5)
 5555555555
  44444444
   333333
    2222
     11
     11
    2222
   333333
  44444444
 5555555555
 5555555555
  44444444
   333333
    2222
     11
```

Function name (9): **`floor_division`**
Parameters: **`original number (int), divisor (int)`**
Return value: **`the maximum number of times the divisor can fully fit in original number (int)`**
Description:
Write a function that takes two integers. The first number will be divided by the second number. Count maximum number of times the divisor can fully fit into the original number. You are essentially coding the floor division operator. **You cannot use the // (that would trivialize this assignment. You must use a loop.**

```
>>> a = floor_division(3,11)
>>> print(a)
>>> 0

>>> b = floor_division(20,6)
>>> print(b)
>>> 3
```

Function name (10): **`multi_table`**
Parameters: **`number to make the multiplication table for (int)`**
Return value: **`N/A`**
Description:
Write a function that takes in a number as a parameter and prints the multiplication table for that number. The table should be printed in an easy to read format. The first row and column should display the numbers from 1 – number specified. **Do not hard code this. You must use a for-loop.**

Here are some test cases to show you how the function should work:

```
>>> multi_table(5)
1       2       3       4       5

2       4       6       8       10

3       6       9       12      15

4       8       12      16      20

5       10      15      20      25
```

Hint: Try using a nested for-loop. Use tab characters to space out the grid properly.

Grading Rubric

- **rearrange_vowels**:	5 points
- **censor**:	5 points
- **odd_and_even_multiples**:	10 points
- **most_common_character**:	10 points
- **sum_multiples**:	10 points
- **remove_vowels**:	10 points
- **guess_dumplings**:	10 points
- **tie**:	15 points
- **count_divisibility**:	10 points
- **multi_table**:	15 points

Provided

The following file(s) have been provided to you. There might be several, but you will only edit:

1. `HW3.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. `HW3.py`