CMSC 201 Fall 2016
Homework 4 – While Loops

Assignment: Homework 4 – While Loops
Due Date: Wednesday, October 5th, 2016 by 8:59:59 PM
Value: 40 points

Collaboration: For Homework 4, collaboration is not allowed – you must work individually. You may still come to office hours for help, but you may not work with any other CMSC 201 students.

Your collaboration statement should state that
`Collaboration was not allowed on this assignment.`

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
# File:    FILENAME.py
# Author:  YOUR NAME
# Date:    THE DATE
# Section: YOUR DISCUSSION SECTION NUMBER
# E-mail:  YOUR_EMAIL@umbc.edu
# Description:
#    DESCRIPTION OF WHAT THE PROGRAM DOES
# Collaboration:
#    COLLABORATION STATEMENT GOES HERE
```

As a reminder, **you are to complete these homeworks only with material that we have covered in class when the homework comes out.** If you are unsure if a method you would like to use is allowed, ask one of the instructors or TAs for clarification. (This means that you cannot use **`for`** loops for this assignment – you may only use **`while`** loops.)

Homework 4 is designed to help you practice using **`while`** loops, branching selection structures, Boolean logic, and printing. More importantly, you will be solving problems using algorithms you create and code yourself.

Remember to enable Python 3 before running and testing your code:

```
scl enable python33 bash
```

Instructions

In this homework, we will be doing a series of exercises designed to make you practice using **`while`** and **`for`** loops, control statements like **`if/else`**, **`print()`** statements, and algorithmic thinking. Each one of these exercises should be in a **separate python file**. For this assignment, you may assume that all the input you get will be of the correct type (*e.g.,* if you ask the user for a whole number, they will give you an integer).

For this assignment, you'll need to follow the class coding standards, a set of rules designed to make your code clear and readable. The class coding standards are on the website, linked at the top of the "Assignments" page. You can also access them directly via this URL (http://goo.gl/yEoGfC).

NOTE: ***You must use*** **`main()`** *as seen in your* **`lab2.py`** *file, and as discussed in class.*

At the end, your Homework 4 files must run without any errors.

Details

Homework 4 is broken up into four separate parts. **Make sure to complete all 4 parts.**

NOTE: Your filenames for this homework must match the given ones exactly.
And remember, filenames are case sensitive!

Questions

Each question is worth the indicated number of points. Following the coding standards, having complete file headers, and having correctly named files is worth 6 points.

hw4_part1.py **(Worth 8 points)**

For this part of the homework you will write code to get valid input from the user. Your program will need to continue to re-prompt them until they provide a username that is at least 4 characters long, and no more than 12 characters long.

Your program must tell the user whether each of their chosen usernames is too short or too long, as well as reminding them of what is an acceptable username length.

Here is some sample output, with the user input in blue.
(Yours does not have to match this exactly, but it should be similar.)

```
bash-4.1$ python hw4_part1.py
Please enter your username: xyz
That username is too short.
The username must be between 4 and 12 characters.
Please enter a new username: maximillian17
That username is too long.
The username must be between 4 and 12 characters.
Please enter a new username: morgan4
Thank you for choosing the username morgan4
```

(HINT: The minimum and maximum lengths should be constants, so that you can easily change them later.)

hw4_part2.py **(Worth 5 points)**

For this part, you are going to write a program that is able to compute modulus **without** using the mod operator.

Your program should ask the user for two numbers, and should compute the answer to **`firstNum % secondNum`**. This should work even if the first number is smaller than the second. Your program should then output the full equation, including the answer, to the user.

For input, you may assume that the user will enter an integer greater than 0.

Here is some sample output, with the user input in blue.
(Yours does not have to match this exactly, but it should be similar.)

```
bash-4.1$ python hw4_part2.py
Please enter the first  number: 15
Please enter the second number: 8
15 % 8 = 7


bash-4.1$ python hw4_part2.py
Please enter the first  number: 33779
Please enter the second number: 8
33779 % 8 = 3
```

(HINT: This sample output doesn't show every possibility, so make sure to do additional testing yourself. Remember that you can always discuss how to test your code with other students, even on "Individual Work Only" assignments like Homework 4.)

hw4_part3.py **(Worth 12 points)**

Next you are going to create an updated, more sophisticated version of the username length checker from Part 1.

*(**WARNING**: This part of the homework is the most challenging, so budget plenty of time and brain power. And read the instructions carefully!)*

For this updated version, your program is going to follow some of the rules of actual UMBC usernames:

1. Usernames cannot be less than two characters long.
2. Usernames cannot be longer than eight characters.
3. Usernames shorter than eight characters must end in a "**1**" (one).
   (UMBC accepts any number, but we've simplified it to just "1".)

Your program must re-prompt the user until they provide a username that satisfies all of the conditions above. It must also tell the user which of the conditions they failed, and how to fix it. (See the sample output for an example of this.)

(HINT: You should be able to build on what you had in Part 1. You should do this program only after completing and fully testing your Part 1 program.)

(HINT: Think carefully about what your conditionals should look like. If necessary, draw a truth table to help figure out what different inputs will do.)

(See the next page for sample output.)

Here is some sample output for hw4_part3.py, with the user input in blue. (Yours does not have to match this exactly, but it should be similar.)

```
bash-4.1$ python hw4_part3.py
Please enter your username: d
Username is too short, it must be at least 2 characters.
Please enter a new username: dd
Usernames under 8 characters must end with a '1'
Please enter a new username: d1
Thank you for choosing the username d1



bash-4.1$ python hw4_part3.py
Please enter your username: short
Usernames under 8 characters must end with a '1'
Please enter a new username: short2
Usernames under 8 characters must end with a '1'
Please enter a new username: notShort
Thank you for choosing the username notShort



bash-4.1$ python hw4_part3.py
Please enter your username: wayTooLong
Username is too long, must be no longer than 8 characters.
Please enter a new username: notLong
Usernames under 8 characters must end with a '1'
Please enter a new username: notLong8
Thank you for choosing the username notLong8
```

(HINT: Pay careful attention when you are reading the sample output. Why is each username accepted at the end?)

hw4_part4.py **(Worth 9 points)**

Next, you are going to write code that simulates the up and down movement of a hailstone in a storm.

Your program should ask the user for a positive integer, which will be the starting height of the hailstone. Based on the current value of the height, you will repeatedly do the following:

- If the current height is 1, quit the program
- If the current height is even, cut it in half (divide by 2)
- If the current height is odd, multiply it by 3, then add 1

Your program will keep updating the number, following the above rules, until the number is 1. You should print out the height of the hailstone at each step. Once the hailstone is at height 1, your program should end, and print out that the hailstone stopped.

(HINT: Think carefully about the order in which your program checks each of the conditions, or it won't perform correctly.)

For example, given a starting value of 24, here are the numbers we output:

```
24 -> 12 -> 6 -> 3 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1
```

(See the next page for sample output.)

Here is some sample output for hw4_part4.py, with the user input in blue. (Yours does not have to match this exactly, but it should be similar.)

```
Please enter the starting height of the hailstone: 36
Hail is currently at height 36
Hail is currently at height 18
Hail is currently at height 9
Hail is currently at height 28
Hail is currently at height 14
Hail is currently at height 7
Hail is currently at height 22
Hail is currently at height 11
Hail is currently at height 34
Hail is currently at height 17
Hail is currently at height 52
Hail is currently at height 26
Hail is currently at height 13
Hail is currently at height 40
Hail is currently at height 20
Hail is currently at height 10
Hail is currently at height 5
Hail is currently at height 16
Hail is currently at height 8
Hail is currently at height 4
Hail is currently at height 2
Hail stopped at height 1
```

(HINT: If you want to prevent your program from outputting decimal numbers like 6.0 and 3.0, you will need to use integer division.)

Submitting

Once your **hw4_part1.py**, **hw4_part2.py**, **hw4_part3.py**, and **hw4_part4.py** files are complete, it is time to turn them in with the **submit** command. (You may turn in individual files as you complete them. To do so, only **submit** those files that are complete.)

You must be logged into your GL account, and you must be in the same directory as your Homework 4 python files. To double-check this, you can type **ls**.

```
linux1[3]% ls
hw4_part1.py  hw4_part2.py  hw4_part3.py  hw4_part4.py
linux1[4]%
```

To submit your Homework 4 python files, we use the **submit** command, where the class is **cs201**, and the assignment is **HW4**. Type in (all on one line) **submit cs201 HW4 hw4_part1.py hw4_part2.py hw4_part3.py hw4_part4.py** and press enter.

```
linux1[4]% submit cs201 HW4 hw4_part1.py hw4_part2.py
hw4_part3.py hw4_part4.py
Submitting hw4_part1.py...OK
Submitting hw4_part2.py...OK
Submitting hw4_part3.py...OK
Submitting hw4_part4.py...OK
linux1[5]%
```

If you don't get a confirmation like the one above, check that you have not made any typos or errors in the command.

You can check that your homework was submitted by following the directions in Homework 0. Double-check that you submitted your homework correctly, since **an empty file will result in a grade of zero for this assignment.**