CMSC 201 Spring 2017
Homework 1 – Pseudocode to Code

Assignment: Homework 1 – Pseudocode to Code
Due Date: Friday, February 10th, 2017 by 8:59:59 PM
Value: 40 points

Collaboration: For Homework 1, collaboration is allowed. Make sure to consult the syllabus about the details of what is and is not allowed when collaborating. You may not work with any students who are not taking CMSC 201 this semester.

If you work with someone, remember to note their name, email address, and what you collaborated on by filling out the Collaboration Log.

You can find the Collaboration Log at http://tinyurl.com/collab201-Sp17.

Remember that all collaborators need to fill out the log each time; even if the help was only "one way" help.

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
```

Instructions

For each of the questions below, you are given a program, expressed in simple pseudocode. (This is similar to the in-class exercises we did during Lecture 2.) From this pseudocode, you must implement a working program in Python. For this exercise, you will only need to use concepts we have discussed in class such as variables, expressions, **`input()`**, casting to an integer, and **`print()`**.

The pseudocode and flowcharts may combine multiple lines of code into one step, or they may split something that would take a single line of code into multiple pieces. Think carefully about what the overall goal of the algorithm is before you begin coding.

At the end, your Homework 1 files must run without any errors.

Additional Instructions – Creating the hw1 Directory

During the semester, you'll want to keep your different Python programs organized, organizing them in appropriately named folders (also known as directories).

For Homework 1, you can store all four of the files you'll be creating in a single directory. First, navigate to the **`Homeworks`** directory inside the **`201`** directory (you can do this in a single "**`cd`**" command, as shown below). Next, create a directory to hold your Homework 1 files, and finally go into it.

```
linux3[1]% cd 201/Homeworks
linux3[3]% mkdir hw1
linux3[4]% cd hw1
linux3[5]%
```

From here, you can use **`emacs`** to start creating and writing your different Homework 1 Python programs.

You don't need to make a separate folder for each file. You should store all four Homework 1 files in the same **`hw1`** folder.

Objective

Homework 1 is designed to help you practice writing and running Python code. You'll also get practice with variables, simple expressions, **`input()`**, casting to an integer, and **`print()`**.

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

```
scl enable python33 bash
```

Task

Some of the programs you write will require you to interact with the user by receiving input from them. Some of the programs will require that you do simple calculations using expressions in Python.

For all four of the programs, you will need to print out information to the user. You will also need to use variables in all four of the programs.

Specification

Prior to this assignment, you should read the Coding Standards, available on Blackboard under "Assignments" and linked on the course website at the top of the "Assignments" page.

For now, you should pay special attention to the sections about:

- Naming Conventions
- Use of Whitespace
- Comments (specifically, File Header Comments)

We will not grade "harshly" on Coding Standards this early in the semester, but you should start forming good habits now. Make sure to pay attention to your TA's feedback when you receive your Homework 1 grade back.

Additional Specifications

For this assignment, you do not need to worry about any "input validation."

If the user enters a different type of data than what you asked for, your program may crash. This is acceptable.

If the user enters "bogus" data (for example: a negative value when asked for a positive number), your program does not need to worry about correcting the value or fixing it in any way.

For example, if your program asks the user to enter a whole number, it is acceptable if your program crashes if they enter something else like "dog" or "twenty" or "88.2" instead.

Here is what that might look like:

```
Please enter a number: twenty
Traceback (most recent call last):
  File "test_file.py", line 10, in <module>
    num = int(input("Please enter a number: "))
ValueError: invalid literal for int() with base 10: 'twenty'
```

Questions

Each question is worth 9 points. Following the directions is worth 4 points.

Question 1

Write your program for Question 1 in a file called **`hw1_part1.py`**.

This program, shown in pseudocode, prints out information about what you (the programmer) ate for breakfast.

Translate this pseudocode into a Python program.

```
Set a number value for a variable called numBacon
Set a number value for a variable called numEggs
Set a number value for a variable called numToast

Print out how many bacon, eggs, and toast you had for
breakfast
Print out the total number of things you ate for breakfast
```

Here is some sample output, using values for *my* breakfast. (The number of bacon, eggs, and toast you eat will probably be different.)

(Yours does not have to match this word for word, but it should be similar.)

```
bash-4.1$ python hw1_part1.py
I had 2 pieces of bacon for breakfast
I had 3 eggs for breakfast
I had 2 pieces of toast for breakfast

I ate a total of 7 things for breakfast
```

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

```
scl enable python33 bash
```

Question 2

Write your program for Question 2 in a file called **`hw1_part2.py`**.

This program, shown in pseudocode, asks for information about how many people are (or are not) wearing a hat, and calculates and prints out the percentage of people who are wearing hats.

Translate this pseudocode into a Python program.

```
Ask the user how many people are wearing hats
Ask the user how many people are not wearing hats
Calculate the percentage of people wearing hats and print it
```

For this program, the names of the variables are not given to you. You should choose meaningful variable names.

Here is some sample output, with the user input in **blue**.

(Yours does not have to match this word for word, but it should be similar. The calculated percentages should be the exact same, given the same input.)

```
bash-4.1$ python hw1_part2.py
How many people are wearing hats? 6
How many people are NOT wearing hats? 10
37.5 % of the people are wearing hats

bash-4.1$ python hw1_part2.py
How many people are wearing hats? 88
How many people are NOT wearing hats? 9
90.72164948453609 % of the people are wearing hats

bash-4.1$ python hw1_part2.py
How many people are wearing hats? 12
How many people are NOT wearing hats? 64
15.789473684210526 % of the people are wearing hats
```

Question 3

Write your program for Question 3 in a file called **`hw1_part3.py`**.

This program, shown in pseudocode, asks the user for the name of their dog, and then prints out that the dog is a good dog.

Translate this pseudocode into a Python program.

```
Ask the user for the name of their dog and store it in a
variable called dogName
Print out that dogName is a good dog
```

Here is some sample output, with the user input in **blue**.

(Yours does not have to match this word for word, but it should be similar.)

```
bash-4.1$ python hw1_part3.py
What is the name of your dog? Spot
Spot is a good dog!

bash-4.1$ python hw1_part3.py
What is the name of your dog? Thor
Thor is a good dog!

bash-4.1$ python hw1_part3.py
What is the name of your dog? His Royal Highness,
Christopher Rupert
His Royal Highness, Christopher Rupert is a good dog!
```

Question 4

Write your program for Question 4 in a file called **`hw1_part4.py`**.

This program, shown in pseudocode, asks the user for information about a castle, and calculates and prints out the total number of rooms.

Translate this pseudocode into a Python program.

```
Ask the user how many floors are in the castle
Ask the user how many rooms are on each floor
Calculate and print the total number of rooms in the castle
```

For this program, the names of the variables are not given to you. You should choose meaningful variable names.

Here is some sample output, with the user input in **blue**.

(Yours does not have to match this word for word, but it should be similar.)

```
bash-4.1$ python hw1_part4.py
How many floors are in the castle? 12
How many rooms are there on each floor? 4
There are 48 rooms in the castle

bash-4.1$ python hw1_part4.py
How many floors are in the castle? 0
How many rooms are there on each floor? 99
There are 0 rooms in the castle

bash-4.1$ python hw1_part4.py
How many floors are in the castle? 678
How many rooms are there on each floor? 452
There are 306456 rooms in the castle
```

Submitting

NOTE: How to submit is covered in detail in Lab 1. If you have not completed Lab 1, you should do so before completing this part of the homework.

Once your **`hw1_part1.py`**, **`hw1_part2.py`**, **`hw1_part3.py`**, and **`hw1_part4.py`** files are complete, it is time to turn them in with the **`submit`** command. (You may also turn in individual files as you complete them. To do so, only **`submit`** those files that are complete.)

You must be logged into your account on GL, and you must be in the same directory as your Homework 1 Python files. To double-check you are in the directory with the correct files, you can type **`ls`**.

```
linux1[3]% ls
hw1_part1.py  hw1_part2.py  hw1_part3.py  hw1_part4.py
linux1[4]%
```

To submit your Homework 1 Python files, we use the **`submit`** command, where the class is **`cs201`**, and the assignment is **`HW1`**. Type in (all on one line) **`submit cs201 HW1 hw1_part1.py hw1_part2.py hw1_part3.py hw1_part4.py`** and press enter.

```
linux1[4]% submit cs201 HW1 hw1_part1.py hw1_part2.py
hw1_part3.py hw1_part4.py
Submitting hw1_part1.py...OK
Submitting hw1_part2.py...OK
Submitting hw1_part3.py...OK
Submitting hw1_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.**