UMBC
AN HONORS UNIVERSITY IN MARYLAND

CMSC 201 Spring 2018
Homework 1 – Pseudocode to Code

Assignment: Homework 1 – Pseudocode to Code
Due Date: Friday, February 15th, 2018 by 11:59:59 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/spring19-201-collab.

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 or float, and **`print()`**.

The pseudocode 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 program 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 six 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[2]% mkdir hw1
linux3[3]% cd hw1
linux3[4]%
```

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 of the Homework 1 files in the same **`hw1`** folder.

Coding Standards

Prior to this assignment, you should read the Coding Standards, 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 must use** `main()` as seen in your `lab2.py` file, and as discussed in class.

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), this is acceptable. (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 error 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 6 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 how many items you (the programmer) might have on your desk.

Translate this pseudocode into a Python program.

```
Set a number value for a variable called numMonitors
Set a number value for a variable called numMice
Set a number value for a variable called numEmptyBottles

Using those variables:
Print out how many monitors, mice, and empty bottles you
have on your desk
Print out the total number of items on your desk
```

Here is some sample output, using sample values for *my* desk. (The number of monitors, mice, and empty bottles you have will probably be different.)

(Yours does not have to match this word for word, but it should be similar. Do not worry about singular vs. plural)

```
linux3[5]% python3 hw1_part1.py
I have 3 monitors on my desk
I have 1 mice on my desk
I have 15 empty bottles on my desk

I have a total of 19 items on my desk
```

Question 2

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

This program, shown in pseudocode, asks for the user's daily income and the cost of their favorite candy bar, and then calculates and prints out how many candy bars their daily income is worth.

Translate this pseudocode into a Python program.

```
Ask the user how much money they make in a day
Ask the user how much their favorite candy bar costs
Calculate the number of candy bars they earned that day 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 costs should be the exact same, given the same input.)

```
linux3[6]% python3 hw1_part2.py
What is your daily income? 50
How much does your favorite candy bar cost? 1.2
You earned 41.66666666666667 candy bars today!

linux3[7]% python3 hw1_part2.py
What is your daily income? 10
How much does your favorite candy bar cost? .8
You earned 12.5 candy bars today!

linux3[8]% python3 hw1_part2.py
What is your daily income? 1
How much does your favorite candy bar cost? 1000
You earned 0.001 candy bars today!
```

Question 3

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

This program, shown in pseudocode, asks the user for their favorite musical instrument, and then prints out that the instrument sounds lovely.

Translate this pseudocode into a Python program.

```
Ask the user for the name of their favorite instrument
and store it in a variable called favoriteInstrument
Print out: The favoriteInstrument sounds lovely!
```

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.)

```
linux3[9]% python3 hw1_part3.py
What is your favorite instrument? Trombone
The Trombone sounds lovely!

linux3[10]% python3 hw1_part3.py
What is your favorite instrument? Cello
The Cello sounds lovely!

linux3[11]% python3 hw1_part3.py
What is your favorite instrument? Trumpet with a
bronze bubble mute
The Trumpet with a bronze bubble mute sounds lovely!
```

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 road trip they are planning, and calculates and prints out the total cost of the trip.

Translate this pseudocode into a Python program.

```
Ask the user the cost of gas
Ask the user the cost of snacks
Ask the user the cost of tolls
Ask the user the cost of driving gloves
Calculate and print the total cost of the trip
```

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 costs should be the exact same, given the same input.)

```
bash-4.1$ python3 hw1_part4.py
What was the cost of the gas? 9.99
What was the cost of the snacks? 15.99
What was the cost of the tolls? 38
What was the cost of the driving gloves? 10.12
The total cost of the trip will be 74.10000000000001

bash-4.1$ python3 hw1_part4.py
What was the cost of the gas? 6.50
What was the cost of the snacks? 9
What was the cost of the tolls? 14.50
What was the cost of the driving gloves? 7.20
The total cost of the trip will be 37.2
```

Question 5

Write your program for Question 5 in a file called **hw1_part5.py**.

This program, shown in pseudocode, asks the user for information about the user's name, and prints out a summary of that information.

Translate this pseudocode into a Python program.

```
Ask the user for their family name, and store it in a
variable called familyName
Ask the user for their first name, and store it in a
variable called firstName
Ask the user for their title, and store it in a variable
called title

Print out: Please rise for firstName of House familyName,
title!
```

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$ python3 hw1_part5.py
What is your family name? Gibson
What is your first name? Katherine
What is your title? Queen of Dog Nation
Please rise for Katherine of House Gibson , Queen of
Dog Nation !

bash-4.1$ python3 hw1_part5.py
What is your family name? Johnson
What is your first name? Benjamin
What is your title? King of Cat Nation
Please rise for Benjamin of House Johnson , King of
Cat Nation !
```

Question 6

Write your program for Question 6 in a file called **hw1_part6.py**.

This program, shown in pseudocode, asks the user for information about a bill they've received, and calculates and prints out how much they should pay in total.

Translate this pseudocode into a Python program.

```
Ask the user how much the bill was
Ask the user what the tax percentage is
Ask the user what percentage they want the tip to be
Calculate and print the total cost of the bill
```

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 cost should be the exact same, given the same input.)

```
bash-4.1$ python3 hw1_part6.py
How much was the bill? 100
How much is tax in your area? 6
What percentage do you want to tip? 15
The total cost of the meal will be 121.0

bash-4.1$ python3 hw1_part6.py
How much was the bill? 26.88
How much is tax in your area? 4.5
What percentage do you want to tip? 16.7
The total cost of the meal will be 32.578559999999996
```

Submitting

NOTE: How to submit is covered in detail in Lab 1 and Homework 0. If you have not read those assignments yet, you should do so before completing this part of the homework.

Once your **hw1_part1.py**, **hw1_part2.py**, **hw1_part3.py**, **hw1_part4.py**, **hw1_part5.py**, and **hw1_part6.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_part3.py  hw1_part5.py
hw1_part2.py  hw1_part4.py  hw1_part6.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 hw1_part5.py hw1_part6.py** and press enter.

```
linux1[4]% submit cs201 HW1 hw1_part1.py hw1_part2.py
hw1_part3.py hw1_part4.py hw1_part5.py hw1_part6.py
Submitting hw1_part1.py...OK
Submitting hw1_part2.py...OK
Submitting hw1_part3.py...OK
Submitting hw1_part4.py...OK
Submitting hw1_part5.py...OK
Submitting hw1_part6.py...OK
linux1[5]%
```

(continued on next page)

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 see what files were successfully submitted by running:
`submitls cs201 HW1`

For more information on how to read the output from **`submitls`**, consult with Homework 0. Double-check that you submitted your homework correctly, since **empty files will result in a grade of zero.**

(OPTIONAL) Advice for Submitting Early and Often:
You can also submit one or two files at a time (a good idea to do as you complete each part, *hint! hint!*) simply by modifying the command. For example, if you wanted to turn in just parts 2 and 3, type in **`submit cs201 HW1 hw1_part2.py hw1_part3.py`** and press enter.

```
linux1[4]% submit cs201 HW1 hw1_part2.py hw1_part3.py
Submitting hw1_part2.py...OK
Submitting hw1_part3.py...OK
linux1[5]%
```

If you're re-submitting, the system will ask that you confirm you want to overwrite each file; make sure that you confirm by typing "y" and hitting enter if you want to do so.

```
linux1[5]% submit cs201 HW1 hw1_part3.py
It seems you have already submitted a file named
hw1_part3.py.
Do you wish to overwrite? (y/n):
y

linux1[6]%
```