CMSC 201Section 60

Homework 3 – Lists and For loops

Due Date: Monday, September 28, by 11:59:59 PM
Value: 50 points

This assignment falls under the standard cmsc201 academic integrity policy. This means you should not discuss/show/copy/distribute your solutions, your code or your main ideas for the solutions to any other student.

Make sure that you have a **complete file header comment at the top of <u>each</u> 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 problem that you must solve or a task you must complete.

You should already be familiar with variables, expressions, **`input()`**, and **`print()`**. You should also be familiar with one-way, two-way, and multi-way decision structures.

This assignment will focus on implementing algorithms using lists and **`for`** loops, including any Boolean logic needed.

NOTE: Your filenames for this homework must match the given ones exactly.

And remember, filenames are case sensitive!

Additional Instructions – Creating the hw3 Directory

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

Just as you did for Homework 1 and Homework 2, you should create a directory to store your Homework 3 files. We recommend calling it **`hw3`**, and creating it inside the **`Homeworks`** directory inside the **`201`** directory.

If you need help on how to do this, refer back to the detailed instructions in Homework 1. (You don't need to make a separate folder for each file. You should store all of the Homework 3 files in the same **`hw3`** folder.)

Allowed Keywords/Built-Ins

For this assignment you should use:

1. For loops, with either range, len
2. Lists, declaration, in keyword, append and remove functions
3. if/elif/else
4. print, string formatting is up to you
5. input
6. casting
7. variable declaration and assignment
8. String operations split(), strip(), concatenation (+)

Coding Standards

Prior to this assignment, you should re-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)
    - ***For Homework 3, you should start using In-Line Comments where appropriate***

Additional Specifications

For this assignment, **you must use** `if __name__ == "__main__":` as discussed in class. Put all of the code that you want to execute in this if statement block.

For this assignment, you do need to worry about "input validation" on a number of the problems. Many of the parts of this assignment center on validating input from the user. For example, the user may enter a negative value, but your program may require a positive value. **Make sure to follow each part's instructions about input validation.** If the user enters a

different type of data than what you asked for, your program may crash. This is acceptable.

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 the indicated number of points. Following the coding standards can be worth up to 2 points per problem. If you do not have complete file headers and correctly named files, you will lose points.

hw3_part1.py - Somebody Feed Pupper (10 points)

1. First you should ask for the number of tasks to perform.
2. Then you should create a list of tasks with that many entries.
3. Ask the user if they have completed each task.
4. Report the tasks that have yet to be completed.

```
linux[0]$ python3 hw3_part1.py
How many tasks do you have? 3
Enter next task: Eat Lunch
Enter next task: Feed Pupper
Enter next task: Write Homework
Here are your tasks:
1. Eat Lunch
2. Feed Pupper
3. Write Homework
Have you completed "Eat Lunch"? (yes/no) no
Have you completed "Feed Pupper"? (yes/no) yes
Have you completed "Write Homework"? (yes/no) no
A remaining task is Eat Lunch
A remaining task is Write Homework

linux[1]$ python3 hw3_part1.py
How many tasks do you have? 0
That's either negative or zero
```

hw3_part2.py - Faulhaber Summing (10 points)

We're going to calculate the Faulhaber sums. The p^{th} Faulhaber sum is:

$$F_p(n) = \sum_{i=1}^{n} i^p = 1^p + 2^p + 3^p + \cdots + (n-1)^p + n^p$$

Note that in this case we need to input two numbers, n and p.

Check to make sure at least that n and p are non-negative.

We then will add up the sum, and display it.

DO NOT USE: sum keyword, importing math (you don't need them anyway).

Sample output on the following page:

```
linux[0]$ python3 hw3_part2.py
What is the power we want to use? -2
How many terms do we want to calculate? -2
Both n and p must be non-negative.

linux[1]$ python3 hw3_part2.py
What is the power we want to use? 5
How many terms do we want to calculate? 17
4767633

linux[2]$ python3 hw3_part2.py
What is the power we want to use? 1
How many terms do we want to calculate? 5
15

linux[3]$ python3 hw3_part2.py
What is the power we want to use? 2
How many terms do we want to calculate? 80
173880

linux[4]$ python3 hw3_part2.py
What is the power we want to use? 17
How many terms do we want to calculate? 5
780248593545
```

hw3_part3.py - **(10 points)**

Make a system which asks first how many steps it should run.

For each step do one of the three things here:

1. "add [name]" - add name to list
2. "remove [name]" remove name from list
    a. if the name is not in the list, inform the user and don't crash
3. "max" output the max length name, if there is a tie, then output the last name in lexicographic order. (you are not permitted to use .sort() or sorted() for this, instead consider using the < or > operators for strings)

Lexicographic means the result of the operator >.

We won't use spaces in the name, so you can use split again.

```
linux[0]$ python3 hw3_part3.py
How many steps should we run? 6
Enter command: add Alex
Alex added.
Enter command: add Samantha
Samantha added.
Enter command: add Jill
Jill added.
Enter command: max
The max name is Samantha
Enter command: remove Samantha
Samantha removed.
Enter command: max
The max name is Jill
```

```
linux[0]$ python3 hw3_part3.py
How many steps should we run? 6
Enter command: add Alex
Alex added.
Enter command: add Samantha
Samantha added.
Enter command: add Jill
Jill added.
Enter command: max
The max name is Samantha
Enter command: remove Samantha
Samantha removed.
Enter command: max
The max name is Jill
```

```
linux[0]$ python3 hw3_part3.py
How many steps should we run? 6
Enter command: add Sam
Sam added.
Enter command: add Kiwi
Kiwi added.
Enter command: add Orange
Orange added.
Enter command: max
The max name is Orange
Enter command: remove Orange
Orange removed.
Enter command: max
The max name is Kiwi
```

hw3_part4.py - List merge **(10 points)**

To "merge" two lists, we want to interleave them, so for instance, if we have the lists [1, 2] and [3, 4] the merge of them will be:

[1, 3, 2, 4]

Another example:

['a', 's', 'd', 'f'] and ['a', 'b', 'c', 'd'] will result in:

['a', 'a', 's', 'b', 'd', 'c', 'f' , 'd']

Write a program that takes in two lists with the same number of elements and then outputs them in the following way.

I want you to output the list itself, don't worry about pretty formatting, because I want to see that the internal lists have in fact been merged into a new list.

DO NOT USE: zip.

You may use:

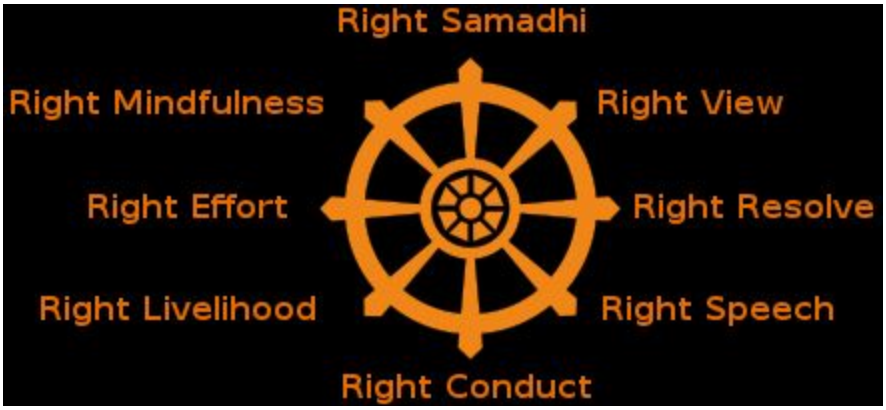
Conditions (if statements), lists (with append and remove), and loops.

```
linux[0]$ python3 hw3_part4.py
How  many elements do you want in each list? 3
What do you want to put in the first list? a
What do you want to put in the first list? b
What do you want to put in the first list? c
What do you want to put in the second list? 1
What do you want to put in the second list? 2
What do you want to put in the second list? 3
The first list is:  ['a', 'b', 'c']
The second list is:  ['1', '2', '3']
The merged list is:  ['a', '1', 'b', '2', 'c', '3']

linux[0]$ python3 hw3_part4.py
How  many elements do you want in each list? 3
What do you want to put in the first list? How
What do you want to put in the first list? you
What do you want to put in the first list? today
What do you want to put in the second list? do
What do you want to put in the second list? feel
What do you want to put in the second list? ?
The first list is:  ['How', 'you', 'today']
The second list is:  ['do', 'feel', '?']
The merged list is:  ['How', 'do', 'you', 'feel', 'today',
'?']
```

hw3_part5.py - Eightfold Path Part II **(10 points)**

We are going to redo the Eightfold Path problem.



This time you MUST use a list to determine the correct result.

You are permitted two if statements, one is
if __name__ == '__main__':
the other one is:
Error checking to ensure that the angle is a multiple of 45.

From the previous assignment:
Note that each point on the wheel represents an angle of 45 degrees apart from the previous angle.

We'll start at zero, the x-axis as usual with Right Resolve, to Right View at 45 degrees, Right Samadhi at 90, Right Mindfulness at 135, etc, and then move counterclockwise as angles go until we reach Right Speech at 315, and finally back to Right Resolve at 360 degrees. It doesn't matter how many times you wind around the wheel, so 360, 720, etc will all lead back to Right Resolve. Negative angles are fine as long as they are divisible by 45.

Your job is to calculate the part of the eightfold path at a particular angle. Use integers, and if you get an integer which is not on the path, you should print "You have not reached enlightenment yet, try an angle divisible by 45."

```
linux3[14]% python3 hw2_part5.py
Enter an angle to determine the point on the Eightfold Path: 121
You have not reached enlightenment yet, try an angle divisible by 45.

linux3[15]% python3 hw2_part5.py
Enter an angle to determine the point on the Eightfold Path: 45
You have selected Right View

linux3[16]% python3 hw2_part5.py
Enter an angle to determine the point on the Eightfold Path: 225
You have selected Right Livelihood

linux3[17]% python3 hw2_part5.py
Enter an angle to determine the point on the Eightfold Path: 0
You have selected Right Resolve

linux3[17]% python3 hw2_part5.py
Enter an angle to determine the point on the Eightfold Path: 720
You have selected Right Resolve

linux3[17]% python3 hw2_part5.py
Enter an angle to determine the point on the Eightfold Path: 405
You have selected Right View

linux3[18]% python3 hw2_part5.py
Enter an angle to determine the point on the Eightfold Path: -45
You have selected Right Speech
```

Submitting

Once your **`hw3_part1.py`**, **`hw3_part2.py`**, **`hw3_part3.py, and hw3_part4.py and hw3_part5.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 3 Python files. To double-check you are in the directory with the correct files, you can type **`ls`**.

```
linux1[3]% ls
hw3_part1.py  hw3_part2.py  hw3_part3.py   hw3_part4.py
hw3_part5.py
linux1[4]%
```

To submit your Homework 3 Python files, we use the **`submit`** command, where the class is **`cmsc201`**, and the assignment is **`HW3`**. Type in the line depicted below and press enter.

```
linux1[4]% submit cmsc201 HW3 hw3_part1.py hw3_part2.py
hw3_part3.py hw3_part4.py hw3_part5.py
Submitting hw3_part1.py...OK
Submitting hw3_part2.py...OK
Submitting hw3_part3.py...OK
Submitting hw3_part4.py...OK
Submitting hw3_part5.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.**