CMSC 201 Fall 2017

Homework 4 – Lists (and More)

Assignment: Homework 4 – Lists (and More)
Due Date: Friday, October 6th, 2017 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.

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 problem that you must solve or a task you must complete.

You should already be familiar with one-way, two-way, and multi-way decision structures. You should also be familiar with **`while`** loops and lists.

This assignment will focus on using lists to store information, as well as using while loops to traverse these lists and decision structures to control the flow of the program.

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

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

Additional Instructions – Creating the hw4 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 previous homeworks, you should create a directory to store your Homework 4 files. We recommend calling it **`hw4`**, 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 4 files in the same **`hw4`** folder.)

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)
- Line Length
- ***Constants***
    - ***For Homework 4, you must use constants instead of magic numbers!!! Magic strings are also forbidden!!!!!!***
- Make sure to **read the last page of the Coding Standards document**, which prohibits the use of certain tools and Python keywords

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 should pay attention to each problem's instructions on using "input validation." 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.

Do note that you do not have to use lists for every part of this homework, and that lists are required for parts 3, 6, and 7 only. However, you may find other parts are easier with the use of lists.

Questions

Each question is worth the indicated number of points. Following the coding standards is worth 4 points. If you do not have complete file headers and correctly named files, you will lose points.

hw4_part1.py (Worth 3 points)

For this part of the homework you will create a "modulus" table.

Your program should prompt the user for two integers. The first integer is the number for which to make a modulus table; the second is how high you want the table to go. You can assume the user will enter integers greater than 0.

The output of the program should be a number, the modulus sign, the number you are modding it by, the equal sign, and the final answer.

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 the number to mod by: 4
Please enter how high you'd like to go: 13
0 % 4 = 0
1 % 4 = 1
2 % 4 = 2
3 % 4 = 3
4 % 4 = 0
5 % 4 = 1
6 % 4 = 2
7 % 4 = 3
8 % 4 = 0
9 % 4 = 1
10 % 4 = 2
11 % 4 = 3
12 % 4 = 0
13 % 4 = 1
```

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

For this part of the homework you will write code to draw a box.

Your program should prompt the user for these inputs, **in exactly this order:**

1. The width of their box
2. The height of their box
3. The symbol the box will be *outlined* in
4. The symbol the box will be *filled* with

For these inputs, you can assume the following:

- The height and width will be positive integers (greater than zero)
- The symbols will be a single character each

Use the *first symbol* to draw a box of the height and width chosen by the user. The box should be filled in with the *second symbol*.

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 width of the box: 8
Please enter the height of the box: 4
Please enter a symbol for the box outline: $
Please enter a symbol for the box fill: -
$$$$$$$$
$------$
$------$
$$$$$$$$

bash-4.1$ python hw4_part2.py
Please enter the width of the box: 1
Please enter the height of the box: 1
Please enter a symbol for the box outline: X
Please enter a symbol for the box fill: O
X
```

HINT: You can keep the `print()` *function from printing on a new line by using* `end=""` *at the end:* `print("Hello", end="")`. *If you do want to print a new line, you can call print without an argument:* `print()`.

hw4_part3.py **(Worth 4 points)**

Create a program that will have the user enter a list of names of the students enrolled in a course. The user can continue entering names indefinitely, stopping only when they enter the sentinel value "QUIT".

Once the user has completed the list, the program should print out the total number of students enrolled in the course. It should then print out whether the course is over enrolled, under enrolled, or perfectly enrolled, based on a desired class size of 8 students. The program **must** make use of a list to accomplish these tasks!

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 hw4_part3.py
Please enter a student name ('QUIT' to stop): Avery
Please enter a student name ('QUIT' to stop): Ben
Please enter a student name ('QUIT' to stop): Nsikan
Please enter a student name ('QUIT' to stop): Kristin
Please enter a student name ('QUIT' to stop): Maria
Please enter a student name ('QUIT' to stop): Emily
Please enter a student name ('QUIT' to stop): Katie
Please enter a student name ('QUIT' to stop): Robert
Please enter a student name ('QUIT' to stop): QUIT
There are 8 students in the course.
The course is perfectly enrolled!

bash-4.1$ python hw4_part3.py
Please enter a student name ('QUIT' to stop): QUIT
There are 0 students in the course.
The course is under enrolled!

bash-4.1$ python hw4_part3.py
Please enter a student name ('QUIT' to stop): quit
Please enter a student name ('QUIT' to stop): stop
Please enter a student name ('QUIT' to stop): exit
Please enter a student name ('QUIT' to stop): QUIT
There are 3 students in the course.
The course is under enrolled!
```

hw4_part4.py **(Worth 6 points)**

Write a program that prompts the user for the number of tests they've taken, and then asks for their grade for each test.

For each test, ask the user (in this exact order),

- If extra credit was allowed ("yes" or "no") on the test.
- What their grade was for the test.

The program must then check the validity of the grade, and re-prompt if the entered grade is invalid. (The program should not re-prompt about extra credit, only for a new grade – see sample output for an example.)

- If extra credit is not allowed, the grade must be between 0 and 100
- If extra credit is allowed, the grade must be 0 or higher

Once the user has entered the information about all of the tests' extra credit status (allowed or not), and a valid score for each test, the program should print out the highest earned grade before exiting.

The program can assume that the number of tests will be at least one.

You are not required to use a list for this part of the homework, but using one may make certain aspects easier.

(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. Using a Boolean flag will also likely make this easier.)

(PRO TIP: The livecoding files posted for Lecture 7 may be a good place to start if you're stuck.)

(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 word for word, but it should be similar.)

```
bash-4.1$ python hw4_part4.py
Enter number of tests taken: 5

For Test # 1 was extra credit allowed?
Please enter 'yes' or 'no': no
Please enter your grade for the test: -1
Test grade must be between 0 and 100 .
Please enter your grade for the test: 101
Test grade must be between 0 and 100 .
Please enter your grade for the test: 100

For Test # 2 was extra credit allowed?
Please enter 'yes' or 'no': no
Please enter your grade for the test: 50

For Test # 3 was extra credit allowed?
Please enter 'yes' or 'no': yes
Please enter your grade for the test: -10
Test grade cannot be negative.
Please enter your grade for the test: 10

For Test # 4 was extra credit allowed?
Please enter 'yes' or 'no': no
Please enter your grade for the test: 9001
Test grade must be between 0 and 100 .
Please enter your grade for the test: 91

For Test # 5 was extra credit allowed?
Please enter 'yes' or 'no': yes
Please enter your grade for the test: 104

The highest grade received was 104
```

hw4_part5.py **(Worth 7 points)**

Write a program that asks the user to enter a PIN number and the type of account it is for, and then checks the PIN for a few different requirements before approving it as secure and repeating the final selection to the user.

The program follows these rules for PIN numbers:

1. If the PIN is for a ***checking*** or ***debit*** account, it must
    a. Be four digits long
        i. This means no leading zeros (0678 is not valid, nor is 0003)
    b. Not end in a "0" (zero)
2. If the PIN is for a ***savings*** account, it must
    a. Be three digits long
        i. This means no leading zeros (035 is not valid, nor is 009)
    b. Not be the same number repeated (333 is not valid, nor is 888, etc.)
        i. *(Think carefully about how to check this one – it's not as difficult as it may seem!)*

The program can assume that the user will enter "c" (for checking), "d" (for debit), or "s" (for savings) as their account type.
The program must re-prompt the user until they provide a PIN/account type combination that satisfies all of the conditions above. It must also tell the user each of the conditions they failed, and how to fix it.
If there is more than one thing wrong (*e.g.*, not four digits, and ends in a zero for a "checking" account), the program must print out both of the things that are wrong/how to fix them.

For this part of the homework, you ***must*** have an in-line comment at the top of **each** of your program's individual `if`, `elif`, and `else` statements, explaining what is being checked by that conditional.

(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. Using a Boolean flag will also likely make this easier.)

(PRO TIP: The livecoding files posted for Lecture 7 may be a good place to start if you're stuck.)

(See the next page for sample output.)

Here is some sample output for **hw4_part5.py**, 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 hw4_part5.py
Please enter a PIN: 6
Please enter the account type (c, d, or s): c
The PIN 6 is too short for a checking account.
Please enter a PIN: 60
Please enter the account type (c, d, or s): d
The PIN 60 is too short for a debit account.
The PIN 60 ends in a zero.
Please enter a PIN: 600
Please enter the account type (c, d, or s): c
The PIN 600 is too short for a checking account.
The PIN 600 ends in a zero.
Please enter a PIN: 6000
Please enter the account type (c, d, or s): c
The PIN 6000 ends in a zero.
Please enter a PIN: 6006
Please enter the account type (c, d, or s): c
Thank you for picking the pin 6006 for your account.

bash-4.1$ python hw4_part5.py
Please enter a PIN: 555
Please enter the account type (c, d, or s): s
The PIN 555 is three numbers repeated.
Please enter a PIN: 55
Please enter the account type (c, d, or s): s
The PIN 55 is too short for a savings account.
Please enter a PIN: 515
Please enter the account type (c, d, or s): s
Thank you for picking the pin 515 for your account.

bash-4.1$ python hw4_part5.py
Please enter a PIN: 6789
Please enter the account type (c, d, or s): s
The PIN 6789 is too long for a savings account.
Please enter a PIN: 6789
Please enter the account type (c, d, or s): d
Thank you for picking the pin 6789 for your account.
```

hw4_part6.py **(Worth 5 points)**

This program allows the user to create a tasks list that will remind them not only of the items on their list, but also the number of hours left to complete each task.
The program must use two separate lists to accomplish this!

The user can continue entering items indefinitely, stopping only when they enter the sentinel value "END". After entering each item, they should be asked how many hours are left before it is completed.

After their list is complete, it should be printed back out to them. Each line must contain the task and the hours left to complete it.

(Your program must NOT prompt for hours after the user enters "END".)

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 hw4_part6.py
Please enter a task, or 'END' to stop: finish homework
Please enter the hour(s) needed to complete it: 6
Please enter a task, or 'END' to stop: submit homework
Please enter the hour(s) needed to complete it: 1
Please enter a task, or 'END' to stop: sleep
Please enter the hour(s) needed to complete it: 8
Please enter a task, or 'END' to stop: nap
Please enter the hour(s) needed to complete it: 2
Please enter a task, or 'END' to stop: eat waffles
Please enter the hour(s) needed to complete it: 1
Please enter a task, or 'END' to stop: END

Here is your task list:
6 hours to complete: finish homework
1 hours to complete: submit homework
8 hours to complete: sleep
2 hours to complete: nap
1 hours to complete: eat waffles
```

hw4_part7.py **(Worth 6 points)**

This program allows the user to create a grocery list, with the user being allowed to enter items indefinitely, stopping only when they enter the sentinel value "STOP". If the user tries to enter an item already in the list, the program should print out an error message and not add the duplicate to the list. **You may NOT use the membership "`in`" operator for this problem.**

After the user indicates they are done, their complete grocery list should be printed back out to them, in the same order it was entered, along with the total number of items on their list.

The program **must** make use of a list to accomplish these tasks!

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 hw4_part7.py
Please enter a grocery item ('STOP' to exit): STOP
There are 0 groceries in your list:

bash-4.1$ python hw4_part7.py
Please enter a grocery item ('STOP' to exit): dog food
Please enter a grocery item ('STOP' to exit): butter
Please enter a grocery item ('STOP' to exit): dog food
Error: The item dog food is already in the list.
Please enter a grocery item ('STOP' to exit): cheese
Please enter a grocery item ('STOP' to exit): dog food
Error: The item dog food is already in the list.
Please enter a grocery item ('STOP' to exit): steak
Please enter a grocery item ('STOP' to exit): ice cream
Please enter a grocery item ('STOP' to exit): steak
Error: The item steak is already in the list.
Please enter a grocery item ('STOP' to exit): STOP
There are 5 groceries in your list:
dog food
butter
cheese
steak
ice cream
```

Submitting

Once your **hw4_part1.py**, **hw4_part2.py**, **hw4_part3.py**, **hw4_part4.py**, **hw4_part5.py**, **hw4_part6.py**, and **hw4_part7.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 4 Python files. To double-check you are in the directory with the correct files, you can type **ls**.

```
linux1[3]% ls
hw4_part1.py  hw4_part3.py  hw4_part5.py  hw4_part7.py
hw4_part2.py  hw4_part4.py  hw4_part6.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 hw4_part5.py hw4_part6.py hw4_part7.py** and press enter.

```
linux1[4]% submit cs201 HW4 hw4_part1.py hw4_part2.py
hw4_part3.py hw4_part4.py hw4_part5.py hw4_part6.py
hw4_part7.py
Submitting hw4_part1.py...OK
Submitting hw4_part2.py...OK
Submitting hw4_part3.py...OK
Submitting hw4_part4.py...OK
Submitting hw4_part5.py...OK
Submitting hw4_part6.py...OK
Submitting hw4_part7.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.**