CMSC 201 Spring 2016
Homework 6 – Nested Loops, While Loops, and Functions

Assignment: Homework 6 – Nested Loops, While Loops, and Functions
Due Date: Monday, March 28th, 2016 by 8:59:59 PM
Value: 40 points

Homework 6 is designed to help you practice using **`while`** loops, nested loops, lists, iterating over lists, branching selection structures, and creating and calling functions. More importantly, you will be solving problems using algorithms you create and code yourself.

Remember to enable Python 3 before you run your programs:

```
     scl enable python33 bash
```

Instructions

In this homework, you will be doing a series of exercises designed to help 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 Blackboard under "Course Documents" in a file titled "CMSC 201 - Python Coding Standards."
You should be commenting your code, and using constants in your code (not magic numbers or strings). Re-read the coding standards!
You will **lose major points** if you do not following the 201 coding standards.

A very important piece of following the coding standards is writing a complete **file header comment block**. Make sure that each file has a comment block at the top (see the coding standards document for an example).

NOTE: **You must use `main()` in each of your files.**

Details

Homework 6 is broken up into three parts. **Make sure to complete all 3 parts.**

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

And remember, filenames are case sensitive.

You must run and test your code before submitting!

Code that does not run will lose a significant number of points.

hw6_part1.py **(Worth 10 points)**

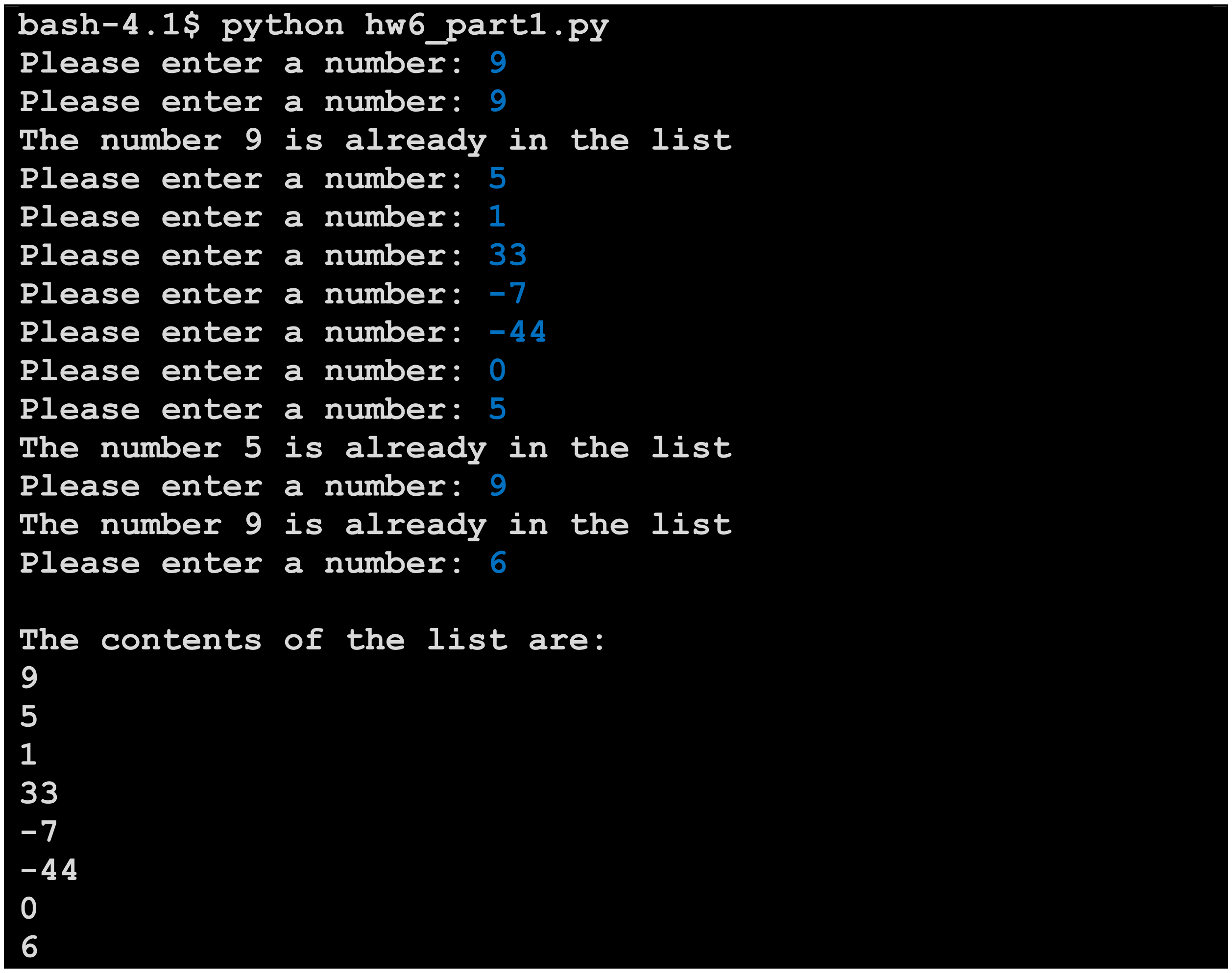
Write a program that reads in numbers from the user and adds them to a list. There should be no duplicate numbers in the list – if the user attempts to add a number already contained in the list, print a short error message and prompt them again.

Once the list contains eight numbers, the program should stop prompting and print out the contents of the list.

You may not use any of Python's other built-in functions. You may not use a line of code that uses the "`in`" keyword to check if a number is already contained in the list. You will lose major points if you do so!
(You should instead use a **`for`** loop to iterate over and check the contents.)

A sample run is available on the following page.

Here is some sample output for **`hw6_part1.py`**, with the user input in blue.



```
bash-4.1$ python hw6_part1.py
Please enter a number: 9
Please enter a number: 9
The number 9 is already in the list
Please enter a number: 5
Please enter a number: 1
Please enter a number: 33
Please enter a number: -7
Please enter a number: -44
Please enter a number: 0
Please enter a number: 5
The number 5 is already in the list
Please enter a number: 9
The number 9 is already in the list
Please enter a number: 6

The contents of the list are:
9
5
1
33
-7
-44
0
6
```

hw6_part2.py **(Worth 10 points)**

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

Next you will write a program that uses a very simple algorithm to compute the prime numbers up to 500.

Your program should start by creating a list of numbers from 2 to 500. (Do **not** hard code a list of 500 number!)

Next, it should remove all of the multiples of 2 from the list, not including 2 (so it should remove 4, 6, 8, 10, etc., but should not remove 2). Remember the **`remove()`** function we covered in lecture!

It should do the same for 3, 4, 5, etc., all the way up to 25. (Again, you should **not** hard code – do not create a separate loop for each of the numbers from 2 to 25!)

Once it is done, you should print the values remaining in the list. (They should all be prime numbers, since we've removed all of the multiples!)

Here is some sample output.

```
bash-4.1$ python hw6_part2.py
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71
73 79 83 89 97 101 103 107 109 113 127 131 137 139 149
151 157 163 167 173 179 181 191 193 197 199 211 223 227
229 233 239 241 251 257 263 269 271 277 281 283 293 307
311 313 317 331 337 347 349 353 359 367 373 379 383 389
397 401 409 419 421 431 433 439 443 449 457 461 463 467
479 487 491 499
```

In the sample output, all of the numbers are printed on the same line. One way to do this yourself is to use something like the line of code below, which replaces the new line that **`print()`** normally puts at the end of each print statement with a single space.

```
    print(number, end = " ")
```

hw6_part3.py **(Worth 16 points)**

Your last task is to write a program that uses functions to draw a triangle, a square, a parallelogram, or all three shapes, as chosen by the user.

Your program should contain four functions:

1. **`main()`**
   - Gets a positive integer (greater than 0) as input from the user
     - Reprompts until the user enters a positive integer
   - Asks the user to choose which shape: “triangle”, “parallelogram”, “square”, or “all” to select all three shapes
     - Reprompts until the user enters a valid choice
   - Calls the appropriate function(s) to print what the user requested
2. **`drawTriangle()`**
   - Takes in an integer as a formal parameter, and draws a solid right-angle triangle (made of asterisks) of that height and width
3. **`drawSquare()`**
   - Takes in an integer as a formal parameter, and draws a solid square (made of asterisks) of that height and width
4. **`drawParallelogram()`**
   - Takes in an integer as a formal parameter, and draws a solid parallelogram (made of asterisks) of that height and width; the parallelogram should be offset by one character for each line

Here is some sample output, with the user input in blue. Additional sample output can be found on the following page.

```
bash-4.1$ python hw6_part3.py
Please enter a positive integer: -1
Please enter a positive integer: -7
Please enter a positive integer: 0
Please enter a positive integer: 5
Please choose the shape: square, parallelogram, triangle,
or all: parallelogram
*****
 *****
  *****
   *****
    *****
```

Additional sample output for **`hw6_part3.py`**, with the user input in blue.

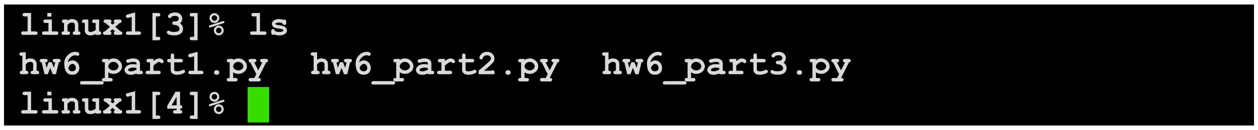
```
bash-4.1$ python hw6_part3.py
Please enter a positive integer: 4
Please choose the shape: square, parallelogram, triangle,
or all: squaree
Please choose the shape: square, parallelogram, triangle,
or all: square
****
****
****
****

bash-4.1$ python hw6_part3.py
Please enter a positive integer: 2
Please choose the shape: square, parallelogram, triangle,
or all: triangle
*
**

bash-4.1$ python hw6_part3.py
Please enter a positive integer: 5
Please choose the shape: square, parallelogram, triangle,
or all: all
*****
*****
*****
*****
*****

*****
 *****
  *****
   *****
    *****

*
**
***
****
*****
```

Submitting

Once all three parts of your Homework 6 are complete, it is time to turn them in with the **`submit`** command.

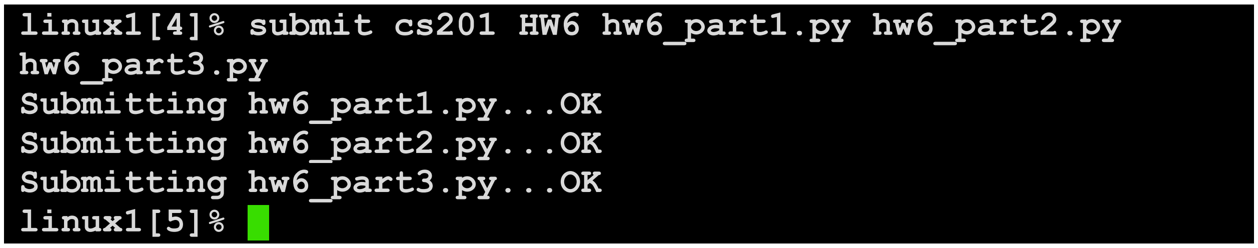
Don't forget to complete the header block comment for each file! Make sure that you updated the header block's file name and description for each file.

You must be logged into your GL account, and you must be in the same directory as the Homework 6 files. To double check this, you can type **`ls`**.



```
linux1[3]% ls
hw6_part1.py  hw6_part2.py  hw6_part3.py
linux1[4]% 
```

To submit your files, we use the **`submit`** command, where the class is **`cs201`**, and the assignment is **`HW6`**. Type in (all on one line)
`submit cs201 HW6 hw6_part1.py hw6_part2.py hw6_part3.py`
and press enter.



```
linux1[4]% submit cs201 HW6 hw6_part1.py hw6_part2.py
hw6_part3.py
Submitting hw6_part1.py...OK
Submitting hw6_part2.py...OK
Submitting hw6_part3.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 **double-check that all three homework files were submitted** by using the **`submitls`** command. Type in **`submitls cs201 HW6`** and hit enter.

And you're done!