# CMSC 201 Fall 2015
## Homework 8 – Recursion

**Assignment:** Homework 8 – Recursion
**Due Date:** Tuesday, November 24th, 2015 by 8:59:59 PM
**Value:** 4% of final grade

Homework 8 is designed to help you further practice using recursion to solve problems. (Remember, a recursive function must have at least one base case and at least one recursive case!)

Remember to enable Python 3 before you run your programs:

```
/usr/bin/scl enable python33 bash
```

## Instructions

Each one of these exercises should be completed 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 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 8 is broken up into three parts. **Make sure to complete all 3 parts. Each part must use a recursive function to solve the given problem.**

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

### hw8_part1.py

For this part of the homework you will write a recursive function that takes in a list of integers as a parameter, and prints out the reverse of that list. Your recursive function should be called **`rev()`**.
The list of integers should be obtained by prompting the user to enter numbers, allowing them to use "**`-1`**" to stop adding integers to the list.

Here is some **sample output** for hw8_part1.py, with the user input in blue.
Your output does not need to be identical, but should be similar.

```
bash-4.1$ python hw8_part1.py
Enter a number to append to the list, or -1 to stop: 2
Enter a number to append to the list, or -1 to stop: 0
Enter a number to append to the list, or -1 to stop: 1
Enter a number to append to the list, or -1 to stop: 7
Enter a number to append to the list, or -1 to stop: 9
Enter a number to append to the list, or -1 to stop: -1
The list as you entered it is:  [2, 0, 1, 7, 9]
The reversed list is:
9
7
1
0
2
```

**hw8_part2.py**

For this part of the homework, you will write a recursive function that prints a right triangle, with the "tip" of the triangle pointing down. Your function should be called **`tri()`**, and must take in two arguments: an integer for the triangle height, and the character the user has chosen to "draw" the triangle with.

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

1. The height of their triangle
2. The symbol the triangle will be made of

**You cannot assume that the provided height will be a valid value!** The user will always provide an integer, but you must perform basic **input validation** to ensure that the height you accept is greater than or equal to 1. Remember to prompt the user with what you will accept as valid input. (Don't just tell them their choice is incorrect; tell them what the acceptable values are as well.)

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

```
bash-4.1$ python hw8_part2.py
Please enter the height of your triangle: -1
Your triangle height must be positive (> 0).
Please enter the height of your triangle: 0
Your triangle height must be positive (> 0).
Please enter the height of your triangle: 8
Please enter a character for your triangle: &

&&&&&&&&
&&&&&&&
&&&&&&
&&&&&
&&&&
&&&
&&
&
```

**hw8_part3.py**
*(WARNING: This part of the homework is the most challenging, so budget plenty of time and brain power.)*

Finally, you will write a program that uses a recursive function to find the greatest common denominator (GCD) of two integers. (The GCD is the largest number that divides evenly into both of the integers. You can see a few examples in the sample output on the next page.)

Your function should be called **`gcd()`**, and should take in three arguments: the first number, the second number, and an integer to help the function "keep track" of its current attempt at finding a GCD value.

*(HINT: Before you tackle the programming part, come up with an algorithm to find the GCD of two integers!)*

Your program should prompt the user for two integers to find the GCD for. Again, **you cannot assume that the provided number will be a valid value!** The user will provide an integer, but you must perform basic **input validation** to ensure that numbers you accept are greater than or equal to 1. Remember to prompt the user with what you will accept as valid input. (Don't just tell them their choice is incorrect; tell them what the acceptable values are as well.)

Sample output for this problem can be found on the next page.

Here is the **sample output** for hw8_part3.py, with the user input in blue. Your output does not need to be identical, but should be similar.

```
bash-4.1$ python hw8_part3.py
Please enter the first integer: -5
Your number must be positive (greater than 0).
Please enter the first integer: 144
Please enter the second integer: 0
Your number must be positive (greater than 0).
Please enter the second integer: 60
The GCD of 144 and 60 is 12

bash-4.1$ python hw8_part3.py
Please enter the first integer: 995
Please enter the second integer: 70
The GCD of 995 and 70 is 5

bash-4.1$ python hw8_part3.py
Please enter the first integer: 37
Please enter the second integer: 121
The GCD of 37 and 121 is 1
```

## Submitting

Once all three parts of your Homework 8 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 8 files. To double check this, you can type **`ls`**.

```
linux1[3]% ls
hw8_part1.py  hw8_part2.py  hw8_part3.py
linux1[4]%
```

To submit your files, we use the **`submit`** command, where the class is **`cs201`**, and the assignment is **`HW8`**. Type in (all on one line) **`submit cs201 HW8 hw8_part1.py hw8_part2.py hw8_part3.py`** and press enter.

```
linux1[4]% submit cs201 HW8 hw8_part1.py hw8_part2.py
hw8_part3.py
Submitting hw8_part1.py...OK
Submitting hw8_part2.py...OK
Submitting hw8_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 HW8`** and hit enter.

And you're done!