CMSC Fall 2016
Homework 0 – Submitting

Assignment: Homework 0 – Submitting
Due Date: Thursday, September 8th, 2016 by 8:59:59 PM
Value: 0 points

Collaboration: For Homework 0, 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 how you collaborated at the top of your file.

If you did not work with anyone else on this assignment, your collaboration statement should state that
I did not collaborate with anyone on this assignment.

If you did work with someone else (or multiple other people), your collaboration statement should state something *similar* to the following, using the other student's name and email, and a short explanation of how they helped you (or you helped them).
I collaborated with Other Student (theirEmail@umbc.edu);
I helped them with something.

Homework 0 is meant to walk you through all of the details of how to submit your CMSC 201 files and check that they were submitted correctly. It is not "worth" any points, but not being familiar with this process could lead to incorrect submissions and a score of 0 on your assignments in this course.

If you run into any errors that you don't understand, email your TA for help, or visit the office hours of any TA or instructor.

Step 1: Navigating to and creating your hw0 directory

Follow the instructions in Lab 1 to log onto the GL system. You are now in your `home` directory. You should have already completed Lab 1, and have already created the `201` directory, and the `Homeworks` and `Labs` directories inside that. The first thing we want to do is go into the `201` directory, and then into the `Homeworks` directory.

```
linux3[1]% cd 201
linux3[2]% cd Homeworks
linux3[3]%
```

If you type "`ls`" at this point, you should see your `hw1` directory that you created in Lab 1. We need to create the directory to hold your Homework 0 file. We'll call the directory hw0, and then we'll go into it.

```
linux3[3]% mkdir hw0
linux3[4]% cd hw0
linux3[5]%
```

To double-check that you are in the right directory, type the "`pwd`" command to see your present working directory (where you "are"). You should be in `home`, then `201`, then `Homeworks`, then `hw0`.

```
linux3[5]% pwd
/afs/umbc.edu/users/k/k/k38/home/201/Homeworks/hw0
linux3[6]%
```

This part of the path will depend on your username.

Step 2: Creating the hw0.py file

Now that you're in the correct directory, it's time to create a simple file. To create a new file, we simply open it for editing using emacs. We'll call this new file **hw0.py**.

```
linux3[6]% emacs hw0.py
linux3[7]%
```

Inside this file, we will want to start off with a "header comment" – we'll learn more about these later, but for now, you can copy and paste the following text, and put in your name, lab section, username, and the collaboration statement that applies to you for this assignment.

```
# File:     hw0.py
# Author:   Your Name
# Date:     9/5/2016
# Section:  Your Lab Section
# E-mail:   your.username@umbc.edu
# Description:
# This file is to help students practice using the
# submission system on the GL servers for CMSC 201.
# Collaboration: Your collaboration statement goes here.
```

You should have a "header comment" similar to this at the top of <u>every</u> file you submit for CMSC 201 this semester.

After you've filled out the header comment, you'll write a single line of Python code.

```
print("This is going to be a great semester!")
```

Once you're done, you'll save by pressing **CTRL+X** and **CTRL+S**. (Press the CTRL and X keys at the same time, then the CTRL and S keys at the same time.) Then exit emacs by pressing **CTRL+X** and **CTRL+C**.

Step 3: Testing the hw0.py file

After you create or update one of your Python files, you should always test that it works correctly. Making one small typo can cause a file to not run the way you want it to, and you wouldn't want to submit a "broken" file.

First, enable Python 3 as shown below. Your prompt should change from "**linux3**" to "**bash-4.1$**" (or something very similar). Once you've switched, run your code to test it. It should print out the message "This is going to be great semester!"

```
linux3[7]% scl enable python33 bash
bash-4.1$ python hw0.py
This is going to be a great semester!
bash-4.1$ █
```

If your program doesn't work at this point, open your **hw0.py** file again, and look closely to see if you made any typos. Fix them, save and exit emacs, and try again. You won't need to "re-enable" Python 3 at this point.

Step 4: Submitting your hw0.py file

Once you are done with your `hw0.py` file, you'll need to submit it to the `cs201` submission directory, and into the correct assignment directory. In this case, the assignment is `HW0`, so that's the one you'll submit to.

```
bash-4.1$ submit cs201 HW0 hw0.py
Submitting hw0.py...OK
bash-4.1$
```

If you've already submitted a file, the system may prompt you to confirm that you want to overwrite your previous submission. If you do want to overwrite your previous file, hit '`y`' for yes.

```
bash-4.1$ submit cs201 HW0 hw0.py
It seems you have already submitted a file named hw0.py.
Do you wish to overwrite? (y/n):
y
Submitting hw0.py...OK
bash-4.1$
```
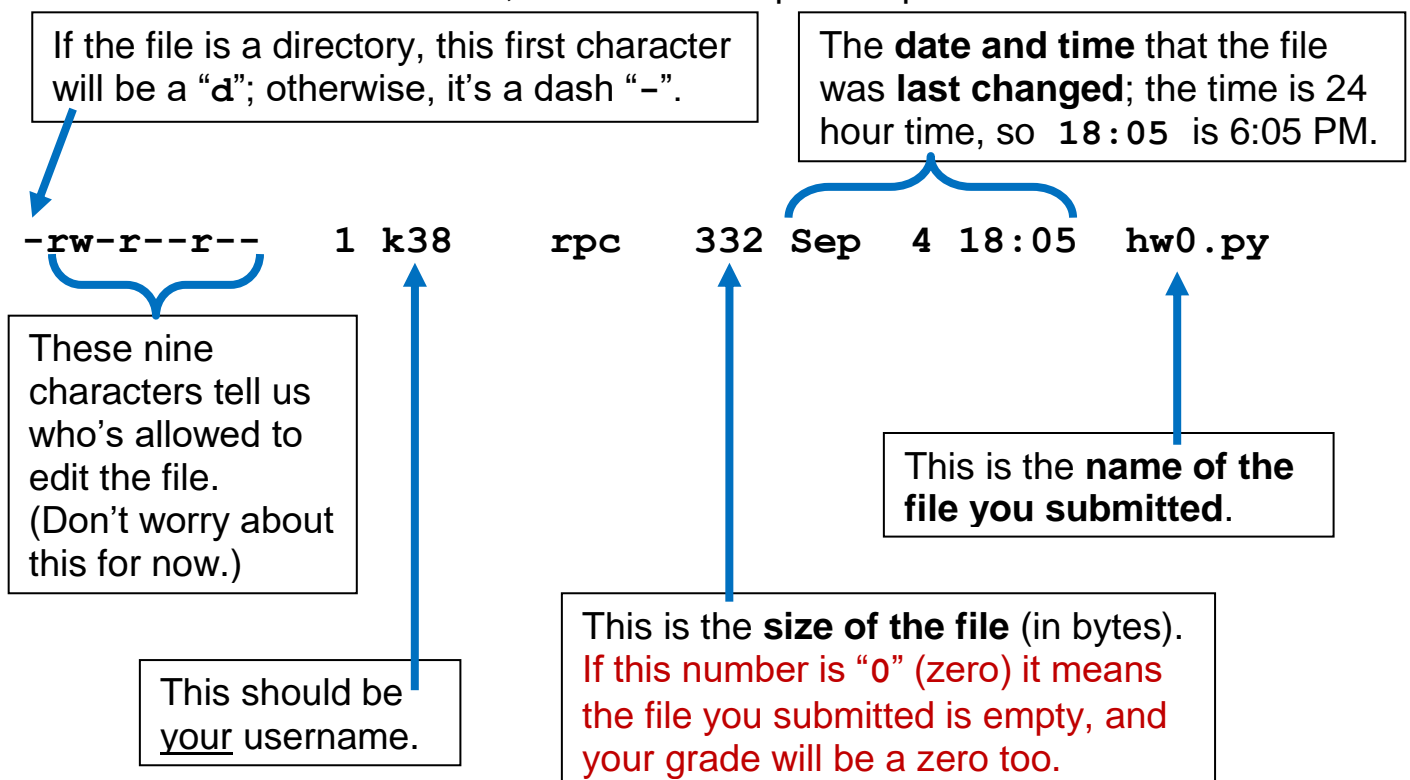
Step 5: Checking your submission

After you submit, you'll need to check that your submission went through, and that you submitted the correct file. We'll start by asking the **submit** system to list all the contents of our submission directory.

```
bash-4.1$ submitls cs201 HW0
total 21
drwx------    2 riehl1 rpc  2048 Sep  4 18:09 .
drwx------ 528 riehl1 rpc 18432 Aug 31 16:54 ..
-rw-r--r--    1 k38    rpc   332 Sep  4 18:05 hw0.py
bash-4.1$ 
```

This gives us a lot of information, so let's break it down. We can ignore the first three lines: the "**total**", and the two folders called "**.**" and "**..**" (one and two periods, respectively). What we care about is any of the remaining lines of text. In this case, because we submitted only one file, there's only one line containing important information; later homeworks may have multiple files, so there will be multiple lines of important information, each about a different file.

Here's the line we care about, and all of its important parts:

If the file is a directory, this first character will be a "**d**"; otherwise, it's a dash "**-**".

The **date and time** that the file was **last changed**; the time is 24 hour time, so **18:05** is 6:05 PM.

-rw-r--r--   1 k38   rpc   332 Sep 4 18:05 hw0.py

These nine characters tell us who's allowed to edit the file. (Don't worry about this for now.)

This should be your username.

This is the **size of the file** (in bytes). If this number is "**0**" (zero) it means the file you submitted is empty, and your grade will be a zero too.

This is the **name of the file you submitted**.

Now let's take a look at the file in <u>your</u> directory, and see if it's the same size. You'll want to ask the system to list the contents, but we're also going to use "`-l`" (dash, and then a lowercase "L") to get a bit more information.

```
bash-4.1$ ls -l
total 2
-rw-r--r-- 1 k38 rpc 332 Sep  4 16:54 hw0.py
-rw-r--r-- 1 k38 rpc 115 Sep  4 14:15 hw0.py~
bash-4.1$ 
```

You should see something like this. If your file was created all in one go, you probably won't have that second file with the "`~`" (tilde) at the end. We'll talk about this file later, but for now let's focus on the "`hw0.py`" line alone. What we want to do is check that the size of the `hw0.py` file in <u>our</u> directory is the same as the one in the <u>submit</u> directory. In this case, both files are `332` bytes, so we know that we have an up-to-date file submitted.

If your submit directory contained an old version, the sizes of the files would be different, and you would want to use the `submit` command again. The dates and times being different are okay, but the size of the file should be the same. (The file in your directory will have the date and time of the last edit; the file in the submit directory has the date and time it was submitted.)

If your file sizes match, then you are done submitting your assignment!

You can use these instructions for any assignment this semester – just replace "`hw0.py`" with the correct file name (or file names) and replace the "`HW0`" submission folder name with the assignment (*e.g.,* `HW2`, `PROJ1`).

If the file you submitted is zero bytes, keep reading.

Troubleshooting: File size of 0 bytes

If the file in your submission folder is showing up as being 0 bytes, the most likely cause is that you accidentally submitted a *directory*, not a file. The submit system doesn't allow you to submit an entire directory, and instead will just create an empty file with the same name.

The empty file problem most often happens when students accidentally submit their "**hw0**" directory, rather than the "**hw0.py**" file. Some students may even accidentally create a "**hw0.py**" *directory*, and then have a "**hw0.py**" *file* inside – this gets very confusing! To check if you've submitted a directory, use the "**ls -l**" command again, and take a close look at the very first character.

```
bash-4.1$ ls -l
total 4
drwxr-xr-x 2 k38 rpc 2048 Sep  4 18:51 Homework0.py
bash-4.1$
```

Do you see the problem with "**Homework0.py**" above? Its first character is a "**d**" instead of a "**-**", which means it's actually a *directory*. If we submitted "**Homework0.py**", the file in the submission directory will be empty (0 bytes).

```
bash-4.1$ submit cs201 HW0 Homework0.py
Submitting Homework0.py...OK
bash-4.1$ submitls cs201 HW0
total 21
drwx------    2 riehl1 rpc  2048 Sep  4 18:53 .
drwx------  528 riehl1 rpc 18432 Aug 31 16:54 ..
-rwxr-xr-x    1 k38    rpc     0 Sep  4 18:53 Homework0.py
bash-4.1$
```

If you've submitted a directory rather than a file, don't worry about deleting it from the submission system (your TA will ignore extra empty files). Instead, you need to find your actual homework file and submit it. It should be "somewhere around here" – depending on how you organized your folders, it may be inside the directory you tried to submit, in the same folder as it, or somewhere else entirely.

First try looking inside that directory, with "`cd hw0`" or "`cd Homework0.py`" or whatever your directory is called, followed by "`ls -l`". If the file is inside that directory, go ahead and follow the instructions from Step 4 and Step 5 again, and you should have a correctly submitted file.

If it's not in that directory, try checking other places in your GL account. If you ever get "lost," you can type "`cd`" (and nothing else), and you will be taken back to your "`home`" directory to start over.

If you run into a problem like this, and you can't figure out how to solve it, go to the TA or instructor office hours for help.

If no one has office hours, you can ask a fellow CMSC 201 student for help! This falls under "Getting or receiving help with using GL or the UMBC Linux system," which is allowed regardless of the collaboration policy for the assignment.
(If the assignment is individual work only, make sure you don't show them any of your code.)

Side Note: The "~" file

Remember that "~" file we saw when we typed "`ls -l`" in Step 5?

```
bash-4.1$ ls -l
total 2
-rw-r--r-- 1 k38 rpc 332 Sep  4 16:54 hw0.py
-rw-r--r-- 1 k38 rpc 115 Sep  4 14:15 hw0.py~
bash-4.1$ █
```

We call that file the "tilde file" (pronounced "til-da") in computer science. If you take a look at the details, you'll notice two things about **hw0.py~** that are different from the "regular" **hw0.py** file.
1. It's an older file: it was last edited at "**14:15**" (2:15 PM), and the regular file was last edited at "**16:54**" (4:54 PM).
2. It's a different size than the regular file. In this case, it's smaller (**115** bytes compared to **332** bytes), but that's not always the case.

The tilde file is a backup of what your work looked like the last time you closed emacs. The emacs program creates it as a service to you, to allow you to see a slightly older version of the file if you want to. You can safely ignore these files – they won't hurt anything.

Side Note: The "#" file
You might also see files that have "**#**" at their beginning and end.

```
bash-4.1$ ls -l
total 2
-rw-r--r-- 1 k38 rpc 334 Sep  4 17:22 #hw0.py#
-rw-r--r-- 1 k38 rpc 332 Sep  4 16:54 hw0.py
bash-4.1$ █
```

These are what emacs creates if your file isn't saved properly (if your GL session crashes, if you choose not to save changes made, etc.). You can safely ignore these files as well. If you do need to recover the file (because GL crashed or you accidentally closed the window), come see a TA or instructor in office hours, and we will walk you through it.