Bridgewater State University – Computer Science Dept.

COMP390 – Software Engineering (Fall 2023)

Prof. Joseph Matta

Individual Coding Assignment 1.1

Overview and Project Requirements

For this assignment, you will add functionality to your meteorite filtering program from Project 1. As in the first project, **NO EXTERNAL LIBRARIES/MODULES ARE ALLOWED. ALSO, NO PYTHON STANDARD LIBRARY IMPORTS! THIS MEANS YOU SHOULD HAVE NO 'IMPORT' STATEMENTS AT THE TOP OF YOUR FILES (EXCEPT FOR IMPORTS OF MODULES YOU WRITE YOURSELF)**. Refer to the Project 1 assignment document for background information about the meteorite filtering program.

In the first meteorite filtering project (Project 1), you filtered the data based on the following parameters:

1) Meteorites that weigh *more than* 2,900,000 grams (*do not* include meteorites that weigh 2,900,000 grams *or less*)
2) Meteorites that fell within the last ten years (*do not* include meteorites that fell *before* 2013. Include meteorites that fell *in 2013 and after*.)

These filter parameters were likely *hardcoded* in Project 1. Also, the filename string identifying the text file containing the data, "meteorite_landings_data.txt," was also probably *hardcoded*. For this project you will improve the meteorite filtering program you wrote for Project 1 by allowing the user to input parameters for filtering either on MASS (g) *or* YEAR. You will also add the ability for the user to specify a target file and the mode with which the file should be opened. For this project, the file should be opened in '***r***' (read) mode because we will not be altering or appending the data text file. However, the user should be given the choice of what mode to open the file in for this version of the project.

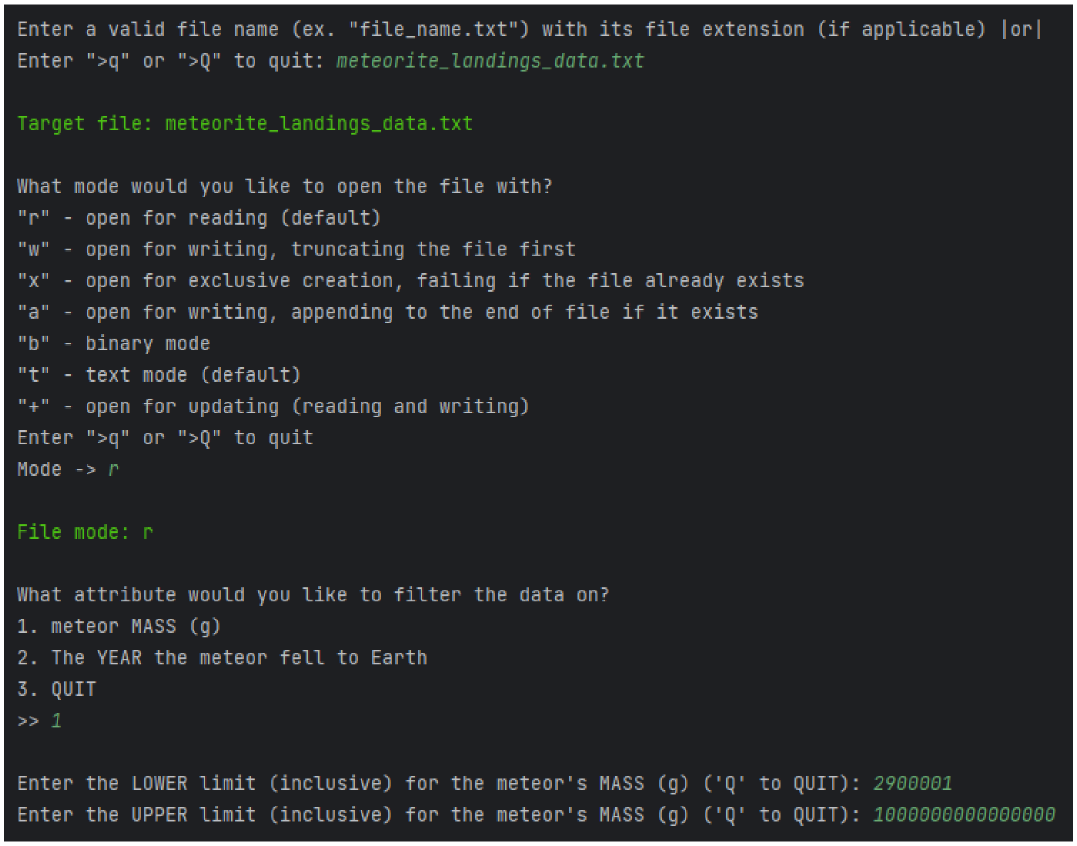
Project 1.1 Requirement Specifics

When the program starts there should be a welcome message printed to the terminal. The exact wording of the message is up to you, but it should give a basic introduction and some developer information. These items MUST be included in the welcome message:

1) A welcome message ("Welcome to the meteorite filtering program…" - for example)
2) Very brief description of the program
3) The developer(s) name (i.e., your name)
4) A release date (usually just the month and year – October 2023)

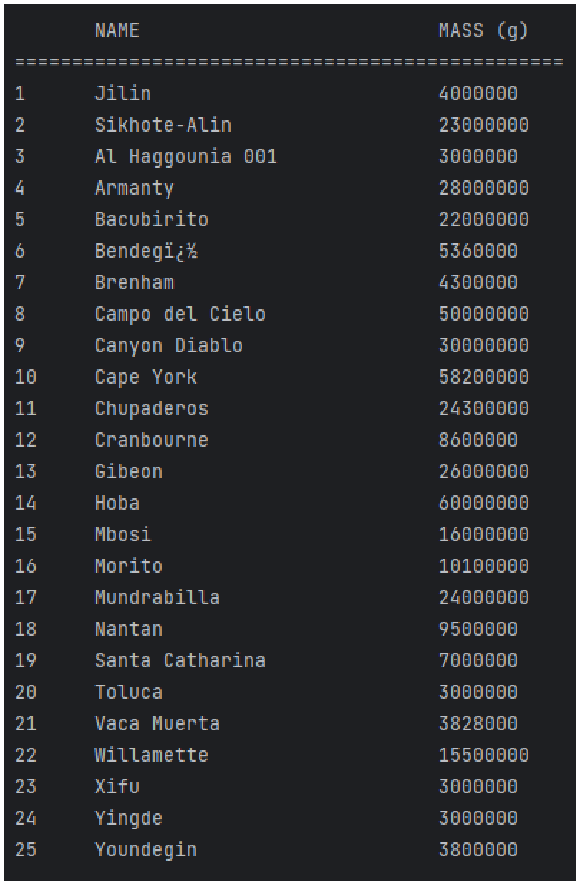
After the welcome message is printed, a series of user input prompts should be displayed in the terminal. Examples of these prompts are shown below.

The wording of your prompts does not need to be *exactly* the same as shown below, however, the prompts must be clear to the user and explicitly communicate what the user needs to do. Thus, your prompts should convey the intended instructions as shown in these examples. **You will lose points if your prompts are not clear and do not explicitly convey the appropriate instructions to the user.** Darker green text at the prompts indicates user input, and the text lines colored in brighter green between the prompt sections are confirmation messages. These confirmation messages **must** be displayed in the output.



```
Enter a valid file name (ex. "file_name.txt") with its file extension (if applicable) |or|
Enter ">q" or ">Q" to quit: meteorite_landings_data.txt

Target file: meteorite_landings_data.txt

What mode would you like to open the file with?
"r" - open for reading (default)
"w" - open for writing, truncating the file first
"x" - open for exclusive creation, failing if the file already exists
"a" - open for writing, appending to the end of file if it exists
"b" - binary mode
"t" - text mode (default)
"+" - open for updating (reading and writing)
Enter ">q" or ">Q" to quit
Mode -> r

File mode: r

What attribute would you like to filter the data on?
1. meteor MASS (g)
2. The YEAR the meteor fell to Earth
3. QUIT
>> 1

Enter the LOWER limit (inclusive) for the meteor's MASS (g) ('Q' to QUIT): 2900001
Enter the UPPER limit (inclusive) for the meteor's MASS (g) ('Q' to QUIT): 1000000000000000
```

The output determined from the user-supplied values shown above at the sample prompts result in the following table being printed to the terminal:



```
       NAME                          MASS (g)
===============================================
1      Jilin                         4000000
2      Sikhote-Alin                  23000000
3      Al Haggounia 001              3000000
4      Armanty                       28000000
5      Bacubirito                    22000000
6      Bendegï¿½                     5360000
7      Brenham                       4300000
8      Campo del Cielo               50000000
9      Canyon Diablo                 30000000
10     Cape York                     58200000
11     Chupaderos                    24300000
12     Cranbourne                    8600000
13     Gibeon                        26000000
14     Hoba                          60000000
15     Mbosi                         16000000
16     Morito                        10100000
17     Mundrabilla                   24000000
18     Nantan                        9500000
19     Santa Catharina               7000000
20     Toluca                        3000000
21     Vaca Muerta                   3828000
22     Willamette                    15500000
23     Xifu                          3000000
24     Yingde                        3000000
25     Youndegin                     3800000
```

This output table should look familiar to you; it's the same table you should have seen in project 1 for the meteorites weighing more than 2,900,000 grams. We got the same output here because the user entered *2,900,001* for the LOWER limit and *1,000,000,000,000,000* (!) for the UPPER limit in the example. Notice that the instructions for entering the limits say "inclusive." This means that the numbers the user enters will be *included* in the output. For example, if there was a meteorite that weighed 2,900,000 grams in the data, it would *not* show up in this table because the user set 2,900,001 as the lower limit. However, if there was a meteorite that weighed 2,900,001 grams, it *would* be included. In other words: "give me all meteorites that have a mass of 2,900,001 *to* 1×10^{15} grams (inclusive on both ends). Make sure to have the user input limits *included* in the filtered results.

Another thing you'll notice is that each prompt has an option to quit the application. You may find it interesting that these "quit" options are executed differently by the user for each set of prompts. For the file access prompts (filename and file mode), the user is required to enter ">q" or ">Q" to quit the application. You may be curious as to why such a strange combination of characters is used to exit the app here. It turns out that there's a good reason for this. If you simply type "Quit" or "Q" at these prompts you may inadvertently open a file named "Quit" or "Q" in the current directory. Although it is unlikely that there is a file named "Quit," we must be very thoughtful when we design our applications.

The “>” character is not allowed in a filename. Therefore, we can be sure that “>Q” and “>q” are not files in the current directory.

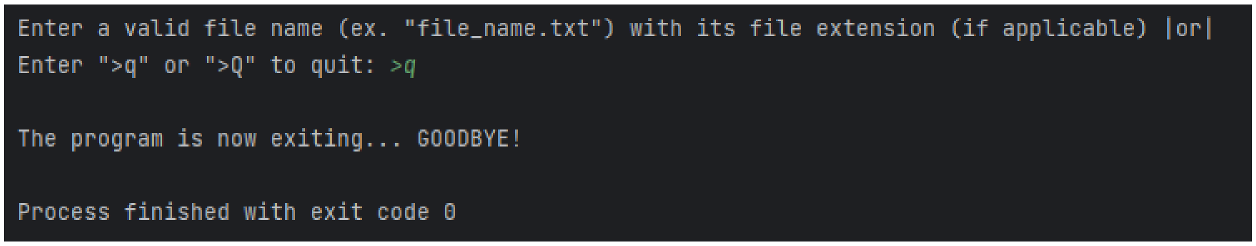
For the *file mode* input, “q” would be accepted as a mode option and will cause an exception. So, again, “>Q” and “>q” are used here.

For the attribute menu, we can simply use a menu option (choice 3) as our exit input.

A simple “q” or “Q” can be used at the LOWER and UPPER limit prompts because we are expecting a number here; the “q” and “Q” characters will not conflict with the expected input for the limits.

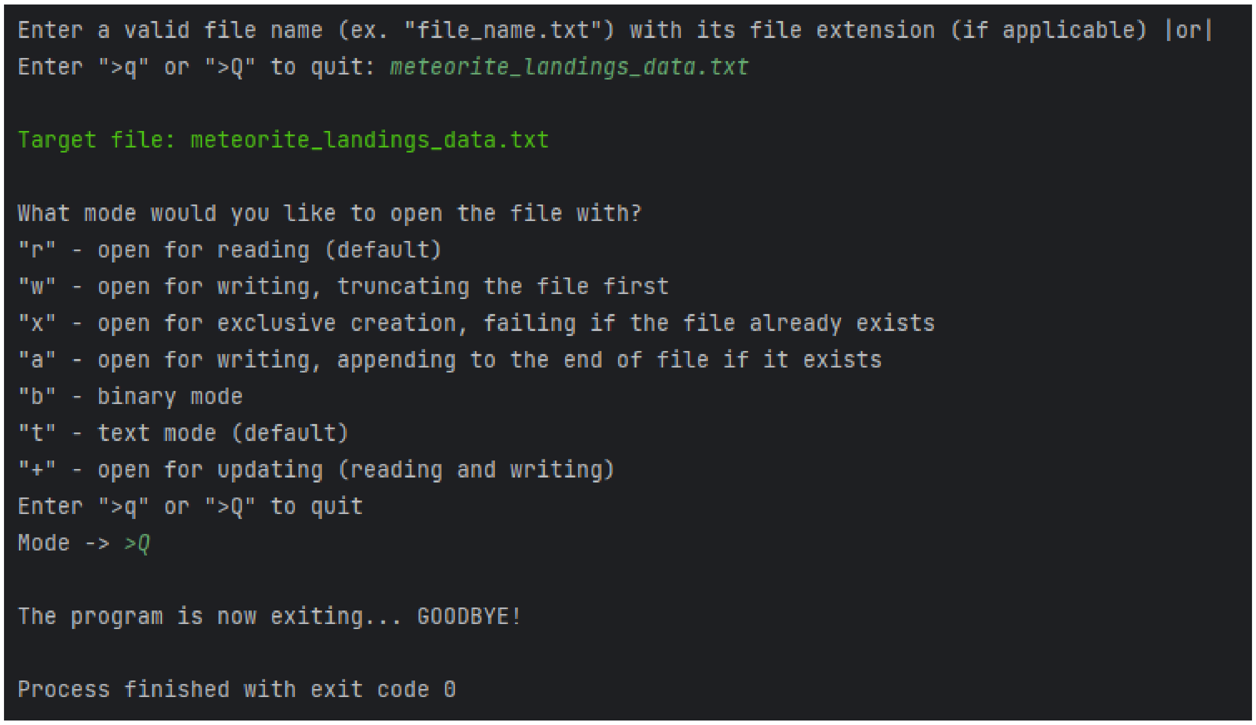
When the user enters the exit option at any of the prompts, the application should immediately exit and close after a short exit message. Examples are shown below. Each panel is a separate program run.

Exiting at the filename input prompt:



```
Enter a valid file name (ex. "file_name.txt") with its file extension (if applicable) |or|
Enter ">q" or ">Q" to quit: >q

The program is now exiting... GOODBYE!

Process finished with exit code 0
```

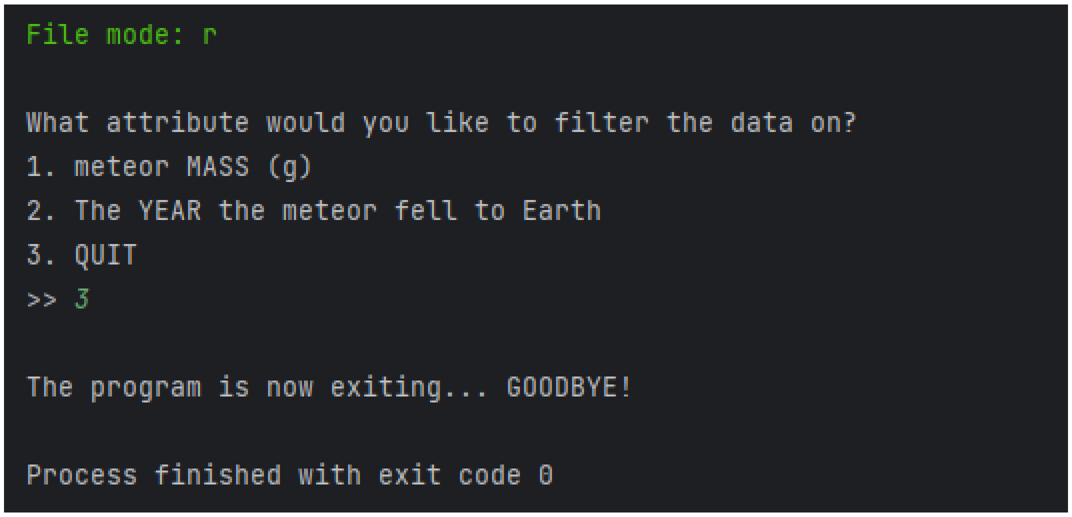
Exiting at the file mode input prompt:



```
Enter a valid file name (ex. "file_name.txt") with its file extension (if applicable) |or|
Enter ">q" or ">Q" to quit: meteorite_landings_data.txt

Target file: meteorite_landings_data.txt

What mode would you like to open the file with?
"r" - open for reading (default)
"w" - open for writing, truncating the file first
"x" - open for exclusive creation, failing if the file already exists
"a" - open for writing, appending to the end of file if it exists
"b" - binary mode
"t" - text mode (default)
"+" - open for updating (reading and writing)
Enter ">q" or ">Q" to quit
Mode -> >Q

The program is now exiting... GOODBYE!

Process finished with exit code 0
```

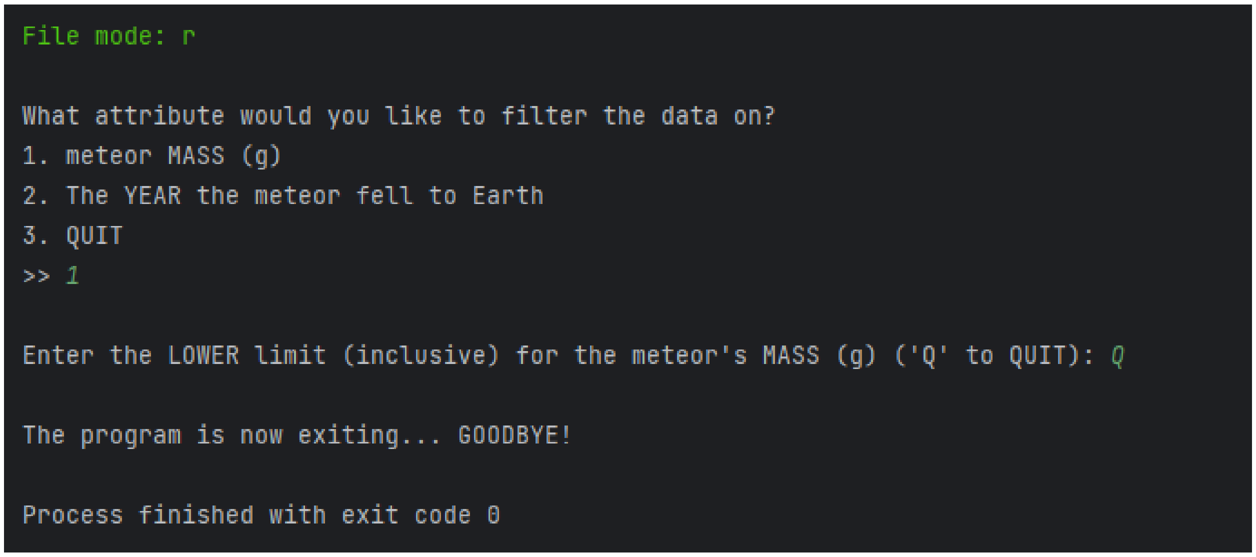
Exiting at the attribute selection menu prompt:

…



```
File mode: r

What attribute would you like to filter the data on?
1. meteor MASS (g)
2. The YEAR the meteor fell to Earth
3. QUIT
>> 3

The program is now exiting... GOODBYE!

Process finished with exit code 0
```

Exiting at the LOWER limit input prompt (this is the same for the UPPER limit prompt):



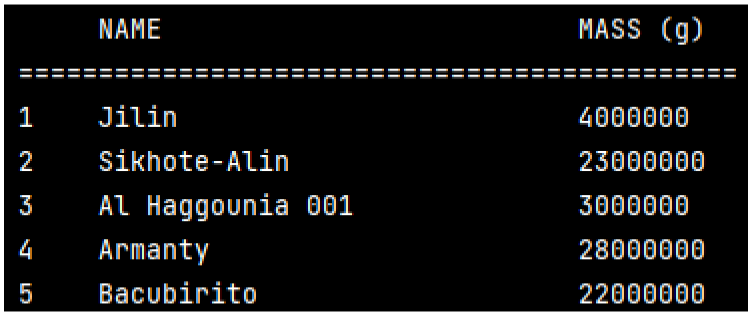
```
File mode: r

What attribute would you like to filter the data on?
1. meteor MASS (g)
2. The YEAR the meteor fell to Earth
3. QUIT
>> 1

Enter the LOWER limit (inclusive) for the meteor's MASS (g) ('Q' to QUIT): Q

The program is now exiting... GOODBYE!

Process finished with exit code 0
```

Your program <u>MUST</u> have this *exit/quit* functionality at each prompt, and there <u>MUST</u> be an exit/GOODBYE message printed before the application exits when the user chooses to exit this way.

Output Requirements

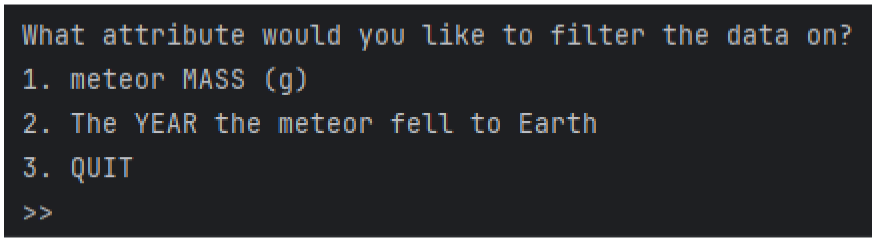
You should output a table to the terminal in the same format as the previous project (Project 1):

The table of filtered data should be **neatly formatted** and printed to the terminal. Each table line should be numbered on the left. A *partial* output example for a NAME/MASS table is shown below. Only five (5) table lines are shown for the MASS table. The complete table may have more *or* less lines depending on the LOWER and UPPER limit parameters input by the user. The purpose of this sample is to show how the table should be formatted. **To receive full credit for this project your output table MUST have the appropriate column titles and include three columns (table line number, meteor name, MASS|YEAR data as shown in the example below.**



```
     NAME                           MASS (g)
=============================================
1    Jilin                          4000000
2    Sikhote-Alin                   23000000
3    Al Haggounia 001               3000000
4    Armanty                        28000000
5    Bacubirito                     22000000
```

For this project, only one table will be printed on a single program run. Unlike Project 1, we are providing the user with a choice: either filter based on MASS (g) *or* YEAR:



```
What attribute would you like to filter the data on?
1. meteor MASS (g)
2. The YEAR the meteor fell to Earth
3. QUIT
>>
```

If the user chooses (1) MASS (g), a mass table will be printed like the one shown in the example above. Alternatively, if (2) is chosen, a YEAR table will be printed. Keep in mind that the table size will vary depending on the LOWER and UPPER limit parameters supplied by the user. The table could have zero (0) rows if the parameters exclude all meteorites, or the table could have 10,000 rows if the parameters define a relatively broad range.

Submission and Additional Requirements

1) Make a 'README.md' file and add it to your project folder. The README **MUST** include the following information (**this is a project requirement!**):
   a. your first and last name
   b. your COMP390 section. For example, if I am in section 001, I would write: COMP390-001
   c. a brief overview of what your project does (don't go deep into implementation details here, but provide a complete and succinct summary of your project)
   d. **a guide on how to use your program.** This section will be a bit extensive because you are adding a significant amount of functionality to your project. Make sure to explain how to use your program using terms and language appropriate for someone who does not have a computer science background. Also, assume that the reader has never used your program before.
   e. a brief description of what does ***not*** work in your project and/or what you did not complete (if you completed all of the project requirements, simply write "all project requirements completed" for this part)
2) Create a new GitHub repository named "COMP390_Individual_Project1_1" (** Make sure your repository is set to PRIVATE **)
3) Push your project to your "COMP390_Individual_Project1_1" repository.
4) Make me (Prof. Matta) a collaborator on your repository. Use my email (j1matta@bridgew.edu) to search for my GitHub account to make me a collaborator. My GitHub username is 'j1matta'. My GitHub profile picture is shown below. Look for this picture when searching for my profile to further confirm that you are indeed making *me* a collaborator and not someone else.

Note: you should push changes made to your project to your GitHub repository often while you are developing your code, not just when you have completed the project. GitHub repositories are a great way to back up your work.

This project is worth 100 points. Please check Blackboard under 'Course Content' for your section's submission deadline. Late submissions will be accepted up to three days past the deadline. However, ten (10) points will be deducted from the total project grade for each day past the submission deadline. **After three days, you will receive zero (0) points for the project.**

~~