Written Narrative

William Haskins

Starting with the software design and engineering artifact, this piece was written in a reverse engineering course where the goal was to create C++ code based off assembly code. Sticking to best programing practices, I made sure to keep the software legible and neat. As part of the project, I was expected to update the main function which I did by providing a method of exiting without first logging in, and by clearing the reader to prevent a variety of input related errors.

During my enhancement of the project, I focused on the readability of the software, updating poorly named variables such as “num1” to something more descriptive. My goal in any project I work on is to make it easy to follow with as few comments as possible, using clean white spacing and descriptive variables.

The databases artifact was a middleware python code between a mongo database and html code which would display information on a user-friendly web page. This code was very functional but also very specific in its usage.

While I do not have access to the mongo database anymore, there were still enhancements that could be made to the python software. I started by renaming the read commands to be more self-descriptive, so the CRUD commands are clearer. Next, I exchanged all the symbolics specific to the database with variables, so that this middleware code could be used with any mongo database so long as the user knows what the database is they wish to access.

Lastly, there is the algorithms and data structure artifact, which was made specifically for this project since I did not have a third suitable artifact to use. I believe one good example of showcasing data structures in programing is by making a set of sorting algorithms, bubble sort and quick sort, that a user could compare against some pre-set lists. This project is broken up into 3 files, the main file, the list selection file, and the sorting file. As usual, I relied on descriptive variables and good white spacing to create legible code.

As this piece was purpose built, there was not too much I could do to enhance it. The largest change was to the input scanner, which before did not close. This could lead to a performance issue in a larger program so I did my best to tackle it by creating a public scanner in the main file, which other files could borrow. This scanner could also be closed in the main function once the user wished to exit the program, removing the risk of a memory leak.