**Project Report 3**

**Problem Statement:**

Homework 3 aims to introduce the concept of recursive binary searching, which is significantly faster than linear searching on average. The tasks include searching for passwords in a file, finding a range of passwords, searching for a specific hash and displaying the corresponding plaintext, and comparing the speed of binary and linear searching methods. The inputs are numerals to navigate the menu and several handpicked passwords most of which are present in the provided text files. Generally, program outputs the password, rank, and hash of searched passwords and, if prompted, a range of passwords. Additionally, the program can search the sorted hash text document to match various hashes, compare binary to linear search, and execute iterative search. Error handling to navigate the menu was required so the program could be used for its main purpose. As all passwords were single tokens, no .ignore was needed for inputs.

**Design:**

Not many alternative paths were taken in completing this project. I coded a binary search, a linear search and an iterative search for the passwords and a separate function for searching for the hashes. The results of the searches were returned in the main function in the cases. Initially, I thought comparing the binary and linear search methods meant analyzing the run time of each, but read closer and found that logging how many indexes were visited was to be the way of comparison. The main data structure used in Homework 3 were vectors containing the sort password text file and sorted hashes text file respectively. The algorithms used are described above and were the core to this project. Nothing I could have done differently would majorly change the results of the program.

**Implementation:**

First, I downloaded and added the various text files to my project. The first real programming I did was creating the Password class. This was like many homework assignments before perhaps even easier as there were only three types of data required to construct the whole class. Nothing new needed to be added to the class. The first function I chose to program was the binary search (Task 3). Once I had that, I could run it twice for the two passwords for Task 2 and make the range a for loop between the passwords. I then coded the linear and iterative search which I am slightly more familiar with. Then came the hash searching which required some conversion but was not anything too different from the password binary search. The code provided were read functions for the text files to fill vectors and be utilized by the program. There was no adaption done to this code. My development timeline was four days.

**Testing:**

The testing process for this assignment was more rigid than the previous. Homework 3 required I enter specific passwords for the password range and various searches with an additional two passwords of my choice as inputs. Normal inputs were various one token passwords concatenating different data types (string, int, char, etc.) this produced a different result depending on whether or not the password was present in the list. Special cases tested had spaces in them which lead to each password not being found as all passwords were only one token. Everything worked as expected.

**Conclusions:**

Homework 3 resulted in a menu-based password searching program that made me feel like a cybersecurity guard. The project was a success with all tests working according to the homework document. I would start working on the project slightly sooner than I did; Turing was being persnickety regarding the warnings and errors not displaying on OnlineGDB. The project took approximately six hours of work to finish but could have been shorter were OnlineGDB not to crash.