This artifact was first created in my data structures and algorithms class. It was a C++ program that uses hash functions to load data, search, add, and remove data. I added this artifact to the ePortfolio because it shows that work that I have done in C++ as well as my use of hash functions. In this database I load the data into the program so that I can search, add, and remove information in the table. There is a clock function that counts how long it takes to perform these functions. I think this program highlights my work with hash algorithms well. The hash function originally used was a division hash function. The time that it took to perform the load function had a huge range from 0.013 seconds to 0.006 seconds. I improved this hash function by making it a double hash function. This closed the range to 0.006 seconds to 0.005 seconds. In a program such as this the amount of time it takes for the data to load may not have a great impact, but on programs that are relying on speed and that are larger in size, the time it takes to load the data is imperative.

The skills that I illustrate are design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution, while managing the trade-offs involved in design choices. The problem that I solved is decreasing the range of time it takes to perform the search, this enhances the program and improves the efficiency of the hash function. In addition, I organized the code so that I am following best coding procedures. Therefore, I have met the course objectives for this artifact.

During the class we performed a lot of algorithms, and we compared the outcome of the time it took to perform the given tasks. But we did not implement double hash functions. Hash functions are very interesting to me because depending on what a programmer is trying to achieve depends on which hash function they should use. This project gave me the opportunity to work with double hash functions and compare its validity to a division hash function. I was forced to think outside the box with this program because my overall goal was to increase the speed of the hash function. The first thing that I tried to use to increase the speed was a multiplication hash function. I quickly realized that based on my data size a multiplication hash function would only slow down the hash function which is also a linear function like division. This led me to try using double hashing. The challenge that I faced was choosing the correct hash function for the data set.