Nick Alvarez

CS202.1101

Project 11

The purpose of this program was to work with vector containers. In doing so, we would learn how to access data within them. The vector would also be templated so any data type could be stored within it, in this case int.

Designing this program allowed some freedom in how the functions could be tackled. Sorting the vector allowed me the possibility of quick sorts, bubble sorts, insertion sorts, and the rest of them. At the end of the day, they would still sort the list, but they would take varying amounts of time because they perform the sort in different ways. Searching also allowed freedom, but in general, the way I implemented it seems to be common.

My biggest issue actually came from the creative freedom of sorting. I originally tried to implement a quick sort. For some god-awful reason, it produced a segmentation fault when I ran it. The for loop within the vector_partition function kept running, and after about two hours of troubleshooting, I decided to go with a bubble sort because I knew how to do that. It made me frustrated because a quick sort is not conceptually difficult, but it was giving me such a hassle.

Improvement could be made in the sort function. I implemented the timer because I was curious to see the difference between my functions and the std library functions. The std::search beat my vector_research function by only around four nanoseconds. The std::sort function, however, beat my vector_resort function by upwards of 70 nanoseconds. I wish I could have properly made the quick sort function instead of a bubble sort so that I could see a more accurate comparison.