1. The results of my program confirmed what I knew by demonstrating that certain functions have better performance than others. For example, bubble sort is a relatively inefficient sorting algorithm when compared to quick sort or insertion sort. Additionally, it confirmed to me that less efficient sorting algorithms are usually easier and simpler to implement, at the cost of performance.

2. The differences that can be seen among slow and quick sorting algorithms is that the slow algorithms require many more comparisons than the quick sorting algorithms. Additionally, they require much more CPU time. For example, on my machine (AMD Ryzen 5 3600 running openSUSE Linux with the ondemand CPU governor) bubble sort took 0.000039 seconds to complete with 50 elements, while merge-insertion sort only took 0.000009 seconds to complete with the same amount of elements, demonstrating that in some cases quick sorting algorithms can be more than 4 times faster. Additionally, quick sorting algorithms are more complex and thus harder to implement. The actual steps are more involved for the programmer, and thus more prone to bugs (I was facing issues with segmentation faults and quick sorting algorithms not working in every run during my development) with the advantage of being dramatically more efficient.