For this project, I created 2 different data structures. I had a map and an unordered map. I used a custom hash function for both data structures, but I made sure to use the same function for both. My hash function (as see below) was a simple one that was provided by Aditya Narasimhan. It takes the first character of the string, converts it to its ascii value, and then mods it with 26.

```cpp
// Hash Function
int hashFunction(string s)
{
    // Write hash function for string using chaining to handle collisions
    // Mod 26
    int hash_value = 0;
    int ascii_value = (int)s[0];
    hash_value = ascii_value % 26;

    return hash_value;
}
```

I ran the program several times, each time increasing the number of insert and remove operations. I did 3 trials for each number of operations, and then took the average. I started with the 10 provided to us with the boilerplate, but then went to 50, then 100, and so on until I had 300 operations. As seen in the graph below, the map was much slower than the unordered map at first. However, once it got to about 200 operations, the time it took to perform all the operations was about the same. At 10 operations, the map was about twice as fast. Then when I performed 200 operations, the time was nearly equal, and this continued until my final round of 300 operations.