For the first part of the project, I coded all the functions and stored them in “operations.c” file. Then in main part I stored all the functions in an array of function pointer. This enabled me to invoke them with a for loop. I created a struct called t_data which holds resulting data. I give this structs pointer to every method and then simply printed the results. I printed result of this parts to “output1.txt” file. Additionally, I sorted the array before I invoked the functions and also, I measured the time elapsed just before I invoke the functions and right after they completed execution.

For the second part of the project I, again, used the struct that I have created as parameters to the threads. We can split this part to two as well. Since I coded my functions in “operations.c” file appropriate for threads for 10 thread I directly used the functions instead of runner functions. I defined my array and its length as global variables so that I didn’t give them as parameters to threads. The results of the operations were recorded to the t_data object. Afterwards I recorded the time elapsed and printed the result to the file named “output2.txt”.

As the other half of this part I, this time, created the runner functions in order to group the functions. This resulted running the project in 5 threads in every thread 2 functions. Then again, I simply recorded the time elapsed and wrote them to the file named “output3.txt”.

My observation for this project was increasing the number of threads doesn’t result in performance gain. I noticed it since the times that are required to execute were almost identical and one threaded one was slightly faster compared to threaded ones. I think this is because of thread creation overhead. Since there is not much of availability to make use of threading for this case overhead was big so that we didn’t notice much of a difference. And also, I can add since the operations that we have performed were efficient algorithms, so it makes it more difficult to observe a gain in threading. To finalize I want to add that I sorted the array before I performed the operations, so it made the operations less costly and made it more difficult to differentiate the performance gain