I tested my program using as many combinations of threads and resources as possible. I haven’t had my program crash yet, and have always terminated the program myself. There really aren't any patterns I've noticed between the variation of combinations other than an increase in threads and resources decreasing the processing time for each thread, meaning that more threads are completing per second as the numbers of thread’s and resources decrease. For the compilation video test I created for the project I tested 10 threads and 10 resources and terminated the program at 10 minutes since I didn’t want the video too long, at this point each thread had run successfully around 200 times. One thing to note about my program is for the output of the requests, I made it so if the output of a certain [thread][resource] is equal zero, I wouldn’t display it and just leave it blank. I did this to make it easier to read the graphs. One thing I did notice is the threads are each generating the same requests in sequence, meaning thread one will get a request, and thread two will get the same request and at first I thought it was an issue with my code, but it turns out to be the fault of the random generator number/seed within C++. Since numbers are being generated so fast and on multiple threads, it only makes sense for them to be generating the same numbers, since the random request generator runs off the internal time clock in the system. But nevertheless, my program works as designed and you can use any combination of threads and resources.