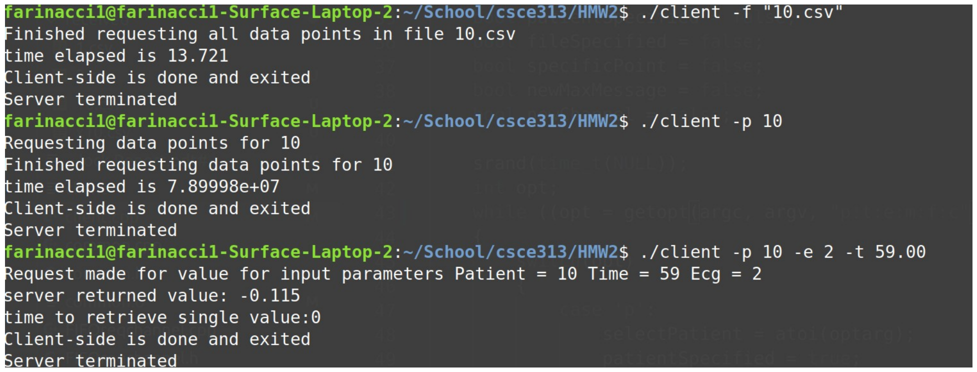
For this assignment I left all files the same only modifying the main function within client.cpp. First I added the getopt function to get user entries from command line when running the executable, which listens for commands “-p -m -t -e -f -c” and initialize related variable and boolean expressions. Following I have some safety checks to determine if inputs are valid. First condition checks that value of p if valid between 1 and number of patients if p is set. Second checks that if either e or t are set that both are set and are in valid file ranges, where valid for t is defined by range 0.00 and 59.996 and a valid e is either 1 or 2. Finally the file is checked, specifically the root of the filename must be correspond to a valid patient number, once again in range 1 and number of patients. If any of these test fail the program terminates.

After the validation steps are done the program proceeds to fork and run the server process. With the server now running the main process continues in the else statement where communication is set between client and server and all requests are handled. There are four major if statements being if(newChannel), if(specificPoint), elseIf(patientSpecified), and if(fileSpecified), where newChannel handles the creation and utilization of new channel, specific point handles the request for a specific point, patientSpecified handles the request for all data-points for single patient, and finally fileSpecified handles the request all data-points for given file.

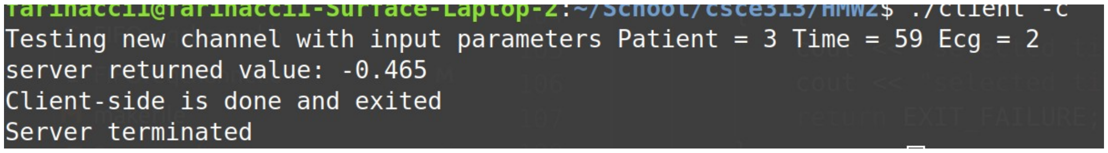
Possible ways to run are:

Create new channel: ./client -c
request point: ./client -p 10 -e 2 -t 59.000
request patient: ./client -p 3
request file: ./client -f “10.csv” -m 5000

specific order does not matter and multiple arguments can be supplied on same line. The only requirement is that if you want all data points for a specific person given by -p <number> you do not supply -e or -t parameters. The assignment specifically requests to handle person 1, but I opted to just have user supply the patient with the format “ ./client -p <number> ” for any patient as its more versatile since it permits testing for any patient and not just a single one.


```
farinacci1@farinacci1-Surface-Laptop-2:~/School/csce313/HMW2$ ./client -f "10.csv"
Finished requesting all data points in file 10.csv
time elapsed is 13.721
Client-side is done and exited
Server terminated
farinacci1@farinacci1-Surface-Laptop-2:~/School/csce313/HMW2$ ./client -p 10
Requesting data points for 10
Finished requesting data points for 10
time elapsed is 7.89998e+07
Client-side is done and exited
Server terminated
farinacci1@farinacci1-Surface-Laptop-2:~/School/csce313/HMW2$ ./client -p 10 -e 2 -t 59.00
Request made for value for input parameters Patient = 10 Time = 59 Ecg = 2
server returned value: -0.115
time to retrieve single value:0
Client-side is done and exited
Server terminated
```

Time is given in milliseconds. As evident requesting the file is exponentially faster than patient as it is filling a buffer and writing that into file as opposed to going value by value.


```
farinacci1@farinacci1-Surface-Laptop-2:~/School/csce313/HMW2$ ./client -c
Testing new channel with input parameters Patient = 3 Time = 59 Ecg = 2
server returned value: -0.465
Client-side is done and exited
Server terminated
```

New channel was created and tested.