Report:

Final part 1:

In order to implement rate I used Poisson distribution that was given. In order to implement bad clients I rescaled the rand that is given in standard library from 0 to 100. If bad = 50 then if random number is less that 50 the client would be slow.

Tests:

To test the distribution I timed clients arrival using 0.5 value meaning 1 client every 2 second. To test how many seconds would take for bigger number of clients to arrive I took 500 client with 40.0 clients per second and the timing was approximately 13 seconds which is close to true value.

Final part 2:

It was completely specified in the documentation.

Final part 3:

I started this part by modifying the producer part, we already have sent size in part and kept socket in the buffer. I added additional read to get “GO\r\n” from server when consumers wants to take an item. I generated data of size BUFSIZE once out of the while loop. And inside the loop I send these chunks to server (chucks size is BUFSIZE). Each time chunk successfully was written I decrement from the size to keep track how much was send when less than BUFSIZE of characters are ready to be sent I sent them separately. The condition to get out of the loop is when characters left to sent is less than 0 or 0.

The next modification was in the server, specifically the place where I handle consumer client. Once item was taken and go for producer written, the server reading characters from producer. It reads chunk of size BUFSIZE keeps it in the allocated memory and sends to consumer immediately. Once server received the characters from producer and send to consumer it exits the loop. The condition is until received characters not equal to declared size of item.

The next modification was regarding consumer. Again the consumer client reads chuck of letters of size BUFSIZE and immediately writes it to dev/null when he received all the possible characters he leaves the loop by the condition when received characters equals to size or bigger. To handle writing to file additional variable ‘error’ was added to keep track of what happened during receiving characters. And then call to ‘createFile’ function was made where I passed error type and number of bytes read before exiting the loop.

Test

To test that my implementation works I ran 500 producers streaming 1Gb of characters. In te server where we keep all the data it would run out of memory fast and crash. To test that writing to file works correctly I ran small amount of clients to check that the size producer declares is the same as consumer writes. I ran 1 file, after making sure that it writes correctly a I ran more clients to see when reject of clients will occur.

Final part 4:

In order to reject slow client I used ‘time’ and ‘difftime’ to get how many seconds user is in the system without identifying himself. I created separate structure where I keep start, end and difference time. I allocated array of size 1000 in the heap for timers. Each timer would be placed in the index of its file descriptor. Once the client arrives which we identify in the accept we start his timer and put it in the array. Once the client identifies himself his timer stops, and when client will get his result from server he will get rid of its timer. After checking which clients identified themselves as producers or consumer I ran loop where I check clients whose timers were not stopped and then I check which of them exceeded REJECT_TIME, besides I do check after timeout in the select expires which is set to 0.5 seconds. I reset it when it reaches value 0. To check that server rejects rights amount of clients I kept track of bad clients in the producer and checked how many rejected. Especially I set bad to 0 and to 100. Note: I added additional boolean field that checks whether the client received produce or consume.

Final part 5:

I added additional condition where I check that the message is ‘STATUS’ and then I checked what exact type of status is asked. To test that it works correctly I ran producers first then check all the fields then ran consumers separately checked all the fields and then both of them to see that everything is consistent. The part 5 helped me to get the mistakes in the code because I saw how many clients exactly were rejected because of timer. Additionally, it helped me understand that I we close client socket only after producers streamed all the characters to not when the producers gave the size of the item. Overall, the part 5 helped me better in debugging an understanding how the whole model of the server works, because before all parts did not connect in a logical sense to me.

Ready state:

My ready state is

./producers localhost 1111 370 2.0 1 &
./consumers localhost 1111 370 2.0 1 &

The moment it almost hits the maximum number of client server manages to serve 240 clients Producers+Consumers. The bufsize is 8192. When rate is changed and more clients arrive, my server is not able to serve all of them without hitting the limit of 512. When I changed the bufsize to 4096 with the same rate an number of clients it also hit the limit my assumption is that transfer takes longer. The empty files might be the result of file creating with an error of writing. Sum of the files might have wrong number of characters because memory allocated by OS to the program server was not enough and not all bytes were able to fit.