Program Design

Overall, the style of programming used was functional programming paradigm.

In addition, I introduced some new functions and made slight adjustments to the original code to allow for more flexibility to make the program easier to code, reliable and as error-free as possible when ran on any machine.

The new functions I introduced were:

1. findLCM(num1, num2) – this function found the LCM of 2 numbers
2. checkRange(num) – checks if the number entered is within the range specified in the project document
3. checkPrime(num) – checks if the number entered is a prime number
4. notPrime() – returns a 'made-up' status message error for when the number entered was not a prime number
5. state (dictionary) – created this dictionary to hold values that needed to be saved during the use of the program

Furthermore, I extensively used the 'processmsg' function to assist in the transmission of the information between client and server.

How does it work?

I requested a connection with the server from the client using the user-inputted information of the IP address and port number. This kicked off the following interaction between the client and server:

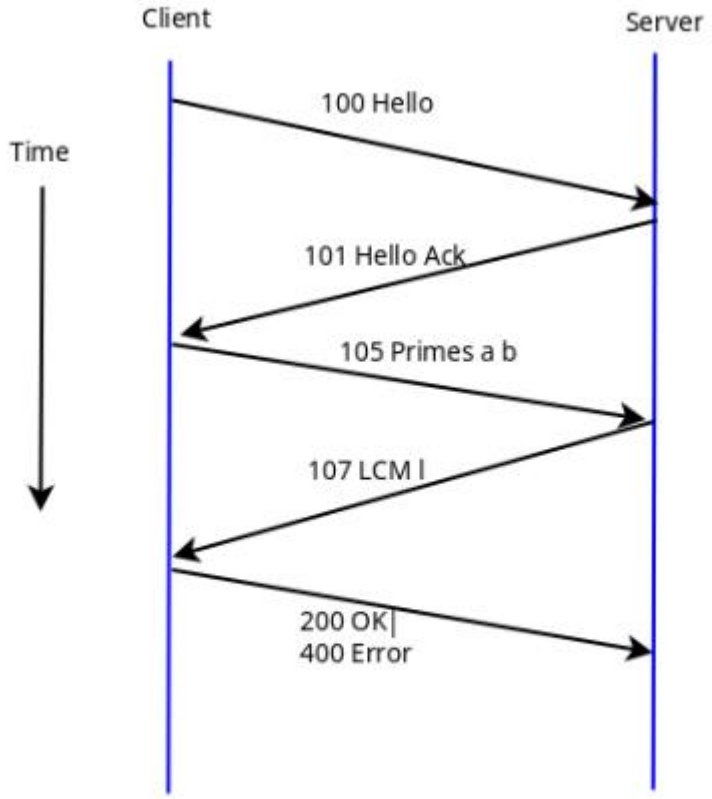

Figure 1: Message sequence exchanged between client and server

Where,

1. On the client side, a socket is created and requests a connection with the desired server
2. After establishing the connection a standard message is sent to 'test' the connection between the client and server, where the client sends a 'Hello' message and the server responds with an acknowledgement.
3. After this standard 'handshake' a LCM calculation request is made, containing prime numbers which is sent from the client to the server
4. The server checks if these numbers are both within range of specification and are prime numbers before it calculates the LCM of both numbers
5. After the calculation, the server responds to the client, the LCM of both numbers
6. The client verifies the LCM from the response of the server with its own calculation of the LCM
7. If the LCMs match then it sends a '200 OK' status message to the server otherwise a '400 Error' is sent

Design Tradeoff(s)

1. In addition to the notPrime function I made it so that if there is a bad request in the form of either non-prime numbers or out of range numbers that a message will be sent to both the client and server to notify the exact issue at hand

2. Made the program work in such a way where multiple messages from the same client can be sent since the server will always stay online as long as the terminal running the python script is open

I made the decision to use TCP and IPv4 addresses for creating the socket for both the client and server as it would be more straightforward and less hectic than UDP counterpart.