Assignment 2 (15% weightage)

Write a POSIX compliant C program(s) for the following scenario. There is a client server application for message passing and logging, which comprises of 1 server and "n" clients. The applications are meant to be running in a single system and exchange data using stateless communication (i.e., single message block should be passed and completed in a single communication loop). Every request from the clients to the server is replied to the client by the server. Every action is initiated by the client and acted upon by the server.

The system follows a request-response mechanism and provides for the following set of actions for the client and server.

The client and server communicate with the following actions

1. **REGISTER**
   This is a client-initiated action where the client connects with the server in the server's **connect channel**, for the very first time with a unique name. The server will verify that the client's name is unique and unused, and return with a key (string). The server will allocate the comm channel, shared memory and create the worker thread for the client.
   The client may disconnect, but the registration will continue to be active.
   The key will be the common shared link for all further communication, which will be used to communicate further on the **comm channel**.

2. **CLIENT_REQUEST**
   The client can send a request to the server, on the comm channel for the client, which will be referenced using the key from the register request for the following actions.
   a. Arithmetic: This action can take the following values

Int N1	Int N2	Int Operation (+, -, *, /)

   b. EvenOrOdd: This action can take the following values

Int N1

   c. IsPrime: This action can take the following values

Int N1

   d. IsNegative: Not supported

   The server will validate the request, with the unique name and key, and the requested operation, and the respond to the client with the result as appropriate.
   The design for this request should be scalable for additional operations.

3. **SERVER_RESPONSE**
   The server response **may be** designed as follows. Students can decide on any alternative schema.

Response Code	Success: 0	Custom Error Code:
Client Response Seq No	The counter for the client, for instance a successful registration will have this value as 1 for each client.	
Server Response Seq No	The counter for the server for the number of responses already served.	
Action Specific Response	To be designed by the students	

   The server will respond to the client's request, by acting on the request.

4. **UNREGISTER**
   At any point of time the client can make this request to the server in the comm channel, which will be acted up on by the server by cleaning up all resources allocated for the client.

The implementation will be based on the following

1. Inter process communication.
   Must be accomplished by using shared memory only.

2. Shared memory
   a. The server will maintain a connect channel for the connect request of clients
   b. The server will create a comm channel for all successful connect request.

   The shared memory will have three segments, with constant sizes for all channels.
   Differential sizing will complicate you code, but with abundance of caution may be used as well.

MUTEX / RD-WR Lock	REQUEST	RESPONSE

   Note the server will have n+1 shared memory objects, where n is the number of registered clients.

3. Synchronization
   The students may use busy-waiting along with a mutex or read write lock or condition variables.
   This must be achieved using pthread mutex in shared process scope. The server will be multithreaded, and each client will be single threaded.

4. The server implementation should be a simple multithreaded process.

5. The client implementation should be an interactive menu driven, with the options of client request and unregister only.
   a. The implementation should provide for single client operations.
   b. The implementation should provide for multiple client operations concurrently.

The following intermediate states of the server/client must be logged.

1. The server initiates and creates the connect channel.
2. The server receives a register request on the connect channel.
   a. On successful creation of a comm channel for a client
   b. On successful response made to the client's register request.
3. The server receives an unregister request
   a. On successful cleanup of the comm channel.
4. The server receives a service request on the comm channel.
5. The server responds to the client on the comm channel.
6. The server should maintain and print the summary info.
   a. The list of registered clients
   b. The count of requests serviced for the client. Note that failed requests which are responded to are also counted in.
   c. The total count of requests serviced for all clients.
7. The client makes register request to the server on the connect channel.
8. The client connects to the server on the comm channel.
9. The client sends a request to the server on the comm channel

10. The client receives a response on the comm channel.
11. All error states, and their mitigations/counter actions.