* PSEUDOCODE

A) TCP/IP Client Side

1. Create a Stream Socket With the socket() call
2. Bind Socket to a local Address with bind (optional)
3. Connect ~~Server~~ socket to a foreign host or Server With Connect().
4. ~~Read~~ Write the message to be sent to the server.
5. Read any message sent by the Server
   (i) Upon sending a message we will receive an Acknowledgement message
   (ii) We will receive other messages sent by other active clients via the server
6. Go to Step 4 untill all data has been exchanged
7. Close Socket and end the TCP/IP session.

B) TCP/IP server side

(2)

(1) Identify Mode of operation from Command Line.
1 - One to One
2 - Broadcast

(2) Bind socket to a local address by using bind(), after creating a socket.

(3) With the listen() call, alert the TCP/IP machine to accept connections

(4) Accept the connection and receive a second socket by using accept(). Store the socket in an array x. and create a thread to handle the particular connection.

(5) If Client i sends a message to Server, it will be handled by Client Thread i.

i) If One to one mode, then send Acknowledgement message to Client i and send the received message to the $(i\%k+1)^{th}$ client where k is the total no. of clients. The message can be sent by looking up the corresponding index in array x which stores the sockets.

ii) If Broadcast Mode, then send Acknowledgement to Client i and send the received message to all the other clients represented by the sockets in array x.

iii) While writing a message to a client use mutex functions to maintain data integrity.
Also join all new Client threads to the main thread.

(6) Go to step 4 to accept other connections.

(7) If a particular client's ~~fd~~ is to be ended, terminate the corresponding thread, and free the socket associated.

① UDP Client Side

1. Create an UDP socket().
2. Send a Test Message to Validate Connection with the server or foreign host.
3. Create a thread which has a function handler designed exclusively to listen for any messages sent by the server by using recvfrom() call, and print it on console.
4. In the main function, ask user to input message to be sent to the server. Repeat step 4 untill all data exchange is finished
5. If all data exchange is finished, then terminate thread, close socket Descriptor and exit.

(1) UDP server side

(1) Identify Mode of Operation from command line argument.
   1 - One to one Mode
   2 - Broadcast Mode

(2) Create an UDP socket, and declare the necessary variable

(3) Bind the socket to a local address by using bind().

(4) Wait for datagram packet containing Test Message to validate connection. on receiving the test message store the sin_port value of the client in an array x in the required index. (Every client has an unique sin_port value).

(5) If received message from client i is not a test message then :

   i) If one to one mode, then send Acknowledgement message to Client i, and send the received message to the $(i\%k+1)^{th}$ client, where k is the total number of Clients. The message can be sent by looking up the corresponding index in array x which contains the sin_port values.

   ii) If Broadcast Mode, then send Acknowledgement to Client i and send the received message to all the other clients represented by sin_port values in Array x.

(6) Go back to step 4 untill all data exchange is finished.

(7) Free the socket descriptor and exit.