1. Implement an UDP client in C to simulate updation of position (in terms of X - Y coordinates of a grid) to a server. Generate these numbers using random(). The client should start with sending its X co-ordinates (-960 to +960) and Y co-ordinates (-640  to 640). The client gets two commands from the server (UPDATE and STOP). UPDATE command contains number of updates N and periodicity r in miliseconds; r between 1 to 1000, N between 1 to 50. After N updates (one every r ms), the client stops and wait for a further command. For STOP command the client should stop sending update.

2.  Implement the UDP server to be able to interact with the client as in (1). It should support multiple clients. The received values from each client is to be stored in an array and the range of movement of each client is to be computed after r updates.

3. Implement an UDP client C program to simulate the receiver of a reliable application. Whenever a packet is received on socket, it should generate a random number between 0 – 1 and drop the packet if the number is <= 0.1, else store it. Packets are of variable length (each packets contains starting byte number and its length in bytes). The client stores each packet and sends an ACK containing next expected byte number. There can be missing  and duplicate packets. It should re-transmit last transmitted ACK if there is no reception during 100 ms. Implement the timer using poll().

4. Implement the UDP server with the complimentary functionality of the client. It transmits packets of variable length every 2 ms, and keep measuring RTT for each acknowledgement of successful transmission. It re-transmits a packet if there are 3 duplicate ACKs for the same packet, or Round-trip timer expires. The timeout value is MAX(100 ms, average of last 10 measured RTT values).

5. Write a TCP client-server program where the client downloads a bytestream of 1 MB or more from the server. Set the MSS in the client side = 576 bytes using setsockopt() function. From tcpdump, find the point where slow start ends and the maximum instantaneous throughput in terms of number of bytes/observed RTT.

6. Write a TCP multi-client server program where the clients send a string to be searched in a file. The name of the file is also mentioned by the client. The server opens the filename mentioned from a specific folder, loads the contents in a large string, searches for the mentioned string, and sends the offset of the position of first location of the string to the requesting client. If the string is not found, it sends -1 to the client. [You can use system() function also to do grep].

7. Write the client program for the server as in (6).