BACKGROUND

This assignment asks you to implement a version of atomic broadcast in a group of processes. Chapter 16 of your notes explains group communication and its various requirements (even if processes crash). For example, in causal order broadcast, if there are two messages m1 and m2 in the system and m1 is ordered before m2, then every non-faulty member must receive m1 before receiving m2. The notes describe how this can be implemented using vector clocks.

WHAT YOU NEED TO DO

Write a program to implement causal order multicast on a distributed system of 16 processes.

You will write a program for a client class. This class will store the name of all the clients along with its own name. This client should also provide the interface including following two functions:

public boolean sendMessage(String strMessage)
   This method should send the passed message to all clients.

public String receiveMessage()
   This method should return the next message for the client from a pool of messages satisfying the causal order requirement. Remember underlying platform is non-FIFO.

public void addClient(String strClientName)
   This method should add another client to this client’s list.
Platform. To communicate between different clients you will use the socket interface designed by TA. It is mostly done to simulate the non-FIFO channels on reliable network and will garble the message order arbitrarily. You will be provided with a class named SimulatedSocket. It has following interface:

```java
public SimulatedSocket(String strClientName)
    This function opens up a new communication channel for the calling client.

public int send(Object objSend, String ReceiverClientName)
    This function sends required object to the recipient client.

public Object receive() -
    This function fetches the next message available for calling client.
    Remember, this message need not be in FIFO order.
```

DELIVERABLES

Specify a distributed chat simulation scenario; (for example, label a message from A to B as (A, B), and its response as Re[1]: (A,B), the response to Re[1]: (A,B) as Re[2]: (A,B) and so on. All messages are sent to the entire group - there are no one-to-one messages. Thus

- Re[3]: (A,B) is causally ordered before Re[5]: (A,B), but
- Re[12]: (A,B) is not causally ordered before Re[2]: (B,C).

You experiment should let the clients run chats with titles as above. At the end of say 32 messages sent by each process, stop the experiment and examine the order of message delivery at each process, and observe that the causal order is not violated.

Please submit the copy of the client class that you have written along with the test program that validates your client class. Both should be separate files, you need not mail class files but java files are must. You will also
submit a small document which will describe why and how your program works.