CO327 – Operating Systems
Lab1: A multi-threaded chat server

Aim: The aim of this laboratory practical is to use threads in developing programs. We will be using the Java programming language for the practical. However, the concepts can be generalized to any programming language. If you unfamiliar with Java threads, you can find more details from the LMS.

Step 0: You are given a simple "ChatServer" code. Download the code via Moodel, compile it and run. Once the chat server is running, a client can connect to the server using some software like *telnet* (or nc on Linux). The server will be listing on the localhost port 666 (see the code).

Depending on the configuration of your system, you might need root permission to run the server which attempts to bind with a socket. Try "sudo java ChatServer".

The server will accept connections from clients and would let clients send in strings over the network. The server will display the string sent by the client and will wait for the next string. When the client sends a message "quit", the server will close the connection with the client and wait for another connection (possibly from a different client) and repeat the same.

Once you have the server running, try to connect with the server twice. You will note that it is not possible to connect with the server twice – in technical terms, concurrent connections are not possible with the server. This is because the current implementation is not multi-thread. There is only one thread in the system. This thread waits for incoming connections and once a clients makes a connection request, the same thread goes and handles the strings which are coming via the socket. So there is no thread – that an instance of execution, to accept new connections.

Laboratory work: Your task is to make this server multi-threaded. While there are number of ways in which you can achieve this, it is recommended to have a single thread waiting for incoming connections and once a socket is created hands over the handling of the socket to new thread.

You can use the code provided to start your implementation and the example code in Moodle demonstrates how threads can be created in Java.

Evaluation: If your server is multi-thread, then it will be capable of handling concurrent connections – more than one client can connect with the system. We will evaluate this basic requirement together with your coding ability.

What to submit: Submit your *ChatServer.java* file via the Moodle submission link. Deadline **5th Feb 2022 @ 2300hr.**