Important Notes:

- This lab will be evaluated automatically. So, please ensure you follow instructions to the dot.
- Do not include unnecessary printf statements. Only print the strings mentioned in question.
- Any strings which are required to be sent by the client or server must be exactly as specified. (This includes the number of bytes that are being sent)
- Certain functions such as those for connection are required for other test cases to run, if not implemented, the latter parts will not be evaluated.
- You'll receive a script containing sample test cases, and your submission will be evaluated against hidden test cases (not the ones you will use for testing) to determine the final score. So refrain from hard coding solutions.
- Submit a zip file with the name <id_no>_lab3.zip containing your implementation of client and server. It should be the same zip provided you with the question.

NOTE:

- To test the first part you need to run the server_tester.py (present in eval folder).
- To test the second part , first you have to create file using create_file.py and then match the file using match_file.py (it will show you the marks)
- Run all the evaluation scripts from ./evals sub-directory.

Welcome to the **Advanced Echo Server Lab**! In this lab, you will be enhancing the capabilities of your echo server program.

- **ECHO COMMAND**: We can create a server command for echoing. When the client wants to echo a message, they just send ECHO:<message>:<number_of_bytes>. The server should parse this command, retrieve the specified message, and send the first n bytes (number_of_bytes) back to the client along with the keyword “OHCE:” added in front of the message. (Assume that <message> does not contain any colon)

  E.g., if client sends ECHO:helloworld:10, then the server responds OHCE:helloworld.
  If the client sends ECHO:helloworld:5, the server responds back with OHCE:hello. (2 marks)
  - If the number of bytes are not specified then it echoes the first 5 bytes as the response message, along with the keyword “OHCE:” added in front. E.g., if client sends ECHO:helloworld, then the server responds OHCE:hello. (Assume that the message length is always greater than 5) (1 marks)
  - If the client message does not contain the command ECHO then the server responds with a message “Invalid command: Unable to echo!”. (1 mark)
  - If the specified number of bytes in the ECHO command exceeds the length of the message, the server will respond by sending the entire message along with the actual length of the message. For instance, if the client sends ECHO:hello:10, the server response would be "OHCE:hello (5 bytes sent)" (without quotes). (2 mark)

    - If the client sends number of bytes as a negative number, then the server sends “Error: Negative number of bytes” as an error message (1 mark)

HINT: Modify the echo input method on the basis of above conditions.

- **Client-Side Logging**: The client-side logging mechanism is designed to facilitate the monitoring of communication between a client and a server. The client creates and maintains a local file named "client_file.txt" to store the communication details in a structured format. Each entry in the file consists of two lines: the first line starts with 'CLIENT: ' followed by the client's message, and the second line starts with 'SERVER: ' followed by the server's response. (3 marks)
    - Example: if the client sends a message ECHO:COMPUTER NETWORKS:8
      Then the file will have following contents:
      CLIENT: ECHO:COMPUTER NETWORKS:8
      SERVER: OHCE:COMPUTER

NOTE: Submit a zip file with the name <id_no>_lab3.zip containing your implementation of client and server. It should be the same zip provided you with the question.