```
//TCP  FileServer.java
import java.io.*;
import java.net.*;

public class FileServer {
    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(5000)) {
            System.out.println("Server is running and waiting for connection...");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected");

                BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

                String fileName = in.readLine(); // Read the file name from client
                System.out.println("Requested file: " + fileName);

                File file = new File(fileName);
                if (file.exists() && !file.isDirectory()) {
                    BufferedReader fileReader = new BufferedReader(new FileReader(file));
                    String line;
                    while ((line = fileReader.readLine()) != null) {
                        out.println(line); // Send each line to client
                    }
                    fileReader.close();
                    out.println("EOF"); // Indicate end of file transfer
                    System.out.println("File sent successfully.");
                } else {
```

```
                out.println("File not found");
                System.out.println("Requested file not found.");
            }
            clientSocket.close(); // Close connection with the client
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
// FileClient.java
import java.io.*;
import java.net.*;

public class FileClient {
  public static void main(String[] args) {
    String serverAddress = "localhost"; // Server address
    int port = 5000;

    try (Socket socket = new Socket(serverAddress, port);
       BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
       PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
       BufferedReader consoleInput = new BufferedReader(new InputStreamReader(System.in))) {

      System.out.print("Enter the file name: ");
      String fileName = consoleInput.readLine();
      out.println(fileName); // transmit  file name to server

      String response;
      System.out.println("File content:");
      while (!(response = in.readLine()).equals("EOF")) {
            System.out.println(response); // Print each line received
          }

        } catch (IOException e) {
          e.printStackTrace();
        }
    }
}
```

CRC

```
public class CRCExample {

    // Method to perform binary division and get the remainder
    public static String divide(String dividend, String divisor) {
        int divisorLength = divisor.length();
        String remainder = dividend.substring(0, divisorLength);

        for (int i = divisorLength; i <= dividend.length(); i++) {
            // If the leftmost bit is '1', XOR with the divisor
            if (remainder.charAt(0) == '1') {
                remainder = xor(remainder, divisor) + (i < dividend.length() ? dividend.charAt(i) : "");
            } else { // If the leftmost bit is '0', XOR with an all-0 divisor
                remainder = xor(remainder, "0".repeat(divisorLength)) + (i < dividend.length() ?
dividend.charAt(i) : "");
            }
            // Remove the leftmost bit to keep remainder at divisorLength-1 bits
            remainder = remainder.substring(1);
        }
        return remainder;
    }
    // XOR operation between two binary strings
    private static String xor(String a, String b) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < a.length(); i++) {
            result.append(a.charAt(i) == b.charAt(i) ? '0' : '1');
        }
        return result.toString();
    }

    public static void main(String[] args) {
        // Given data and divisor
        String data = "1001";
        String divisor = "1011";

        // Append zeros to the data (equivalent to the length of the divisor minus 1)
        String dividend = data + "0".repeat(divisor.length() - 1);

        // Calculate CRC remainder
        String remainder = divide(dividend, divisor);
        System.out.println("CRC Remainder: " + remainder);

        // Display the transmitted data with CRC appended
        String transmittedData = data + remainder;
        System.out.println("Transmitted Data with CRC: " + transmittedData);
    }
}
```

```
CRC Remainder: 101
Transmitted Data with CRC: 1001101
```

3.Sliding window

```
import java.util.LinkedList;
import java.util.Queue;

public class Sender {
    private static final int TOTAL_PACKETS = 10;
    private static final int WINDOW_SIZE = 4;

    private int base;  // The lowest packet number in the sender's window
    private int nextSeqNum;  // The next packet number to be sent
    private Queue<Integer> window;  // Represents the sliding window

    public Sender() {
        base = 0;
        nextSeqNum = 0;
        window = new LinkedList<>();
    }

    public void sendPackets() {
        while (base < TOTAL_PACKETS) {
            // Send packets in the window
            while (nextSeqNum < base + WINDOW_SIZE && nextSeqNum < TOTAL_PACKETS) {
                System.out.println("Sender: Sending packet " + nextSeqNum);
                window.add(nextSeqNum);  // Add packet to the window
                nextSeqNum++;
            }

            // Simulate receiving acknowledgments
            int ack = Receiver.receiveAck(window);
            if (ack >= base) {
                System.out.println("Sender: Received ACK for packet " + ack);
                base = ack + 1;  // Slide the window
                window.removeIf(packet -> packet <= ack);  // Remove acknowledged packets
            }
        }

        System.out.println("Sender: All packets sent and acknowledged.");
    }

    public static void main(String[] args) {
        Sender sender = new Sender();
        sender.sendPackets();
    }
}
```

Receiver.java

```
import java.util.Queue;

public class Receiver {
    private static int expectedPacket = 0;  // The next expected packet number

    public static int receiveAck(Queue<Integer> window) {
        for (Integer packet : window) {
            // Simulate receiving the packet
            if (packet == expectedPacket) {
                System.out.println("Receiver: Received packet " + packet);
                expectedPacket++;
                // Send acknowledgment
                return packet;
            } else {
                System.out.println("Receiver: Unexpected packet " + packet + ". Waiting for packet " +
expectedPacket);
```

```
            }
        }
        return -1;  // Return -1 if no packets were acknowledged
    }
}
```

****javac Sender.java Receiver.java   ***java Sender