9.Write a code simulating RARP protocol for the server and client. write in java code.

Server

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class RARPServer {

   private static final int PORT = 5570;

   public static void main(String[] args) {
      try {
         DatagramSocket socket = new DatagramSocket(PORT);

         System.out.println("RARP Server is running...");

         while (true) {
            byte[] receiveBuffer = new byte[1024];

            // Receive RARP request from the client
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
            socket.receive(receivePacket);

            // Extract hardware address from the request
            String hardwareAddress = new String(receivePacket.getData(), 0,
receivePacket.getLength());
            System.out.println("Received RARP request for hardware address: " + hardwareAddress);

            // Simulate RARP resolution (In this example, a simple mapping is used)
            String ipAddress = resolveIpAddress(hardwareAddress);

            // Send the simulated RARP response back to the client
            byte[] sendBuffer = ipAddress.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length,
receivePacket.getAddress(), receivePacket.getPort());
            socket.send(sendPacket);
         }
      } catch (IOException e) {
         e.printStackTrace();
      }
   }

   private static String resolveIpAddress(String hardwareAddress) {
      // In a real RARP server, this method would perform actual RARP resolution.
      // For simplicity, we use a simple mapping here.
      switch (hardwareAddress.toLowerCase()) {
```

```
            case "00:0a:95:9d:68:16":
                return "192.168.1.1";
            case "00:0a:95:9d:68:17":
                return "192.168.1.2";
            default:
                return "Unknown IP Address";
        }
    }
}
```

Client

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;

public class RARPClient {

    private static final String SERVER_IP = "localhost";
    private static final int SERVER_PORT = 5570;

    public static void main(String[] args) {
        try {
            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter hardware address for RARP request: ");
            String hardwareAddress = scanner.nextLine();

            // Send RARP request to the server
            DatagramSocket socket = new DatagramSocket();
            InetAddress serverAddress = InetAddress.getByName(SERVER_IP);

            byte[] sendBuffer = hardwareAddress.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length,
serverAddress, SERVER_PORT);
            socket.send(sendPacket);

            // Receive the simulated RARP response from the server
            byte[] receiveBuffer = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
            socket.receive(receivePacket);

            // Extract the IP address from the response
            String ipAddress = new String(receivePacket.getData(), 0, receivePacket.getLength());
            System.out.println("Simulated RARP resolution result: " + ipAddress);
```

```
            // Close the socket
            socket.close();

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

Output

RARP Server Output:

RARP Server is running...

Received RARP request for hardware address: 00:0a:95:9d:68:16

RARP Client Output:

Enter hardware address for RARP request: 00:0a:95:9d:68:16 Simulated RARP resolution result: 192.168.1.1

1.Write a program with input " hello_client" , the server listens for, and accepts, a single TCP connection; it reads all the data it can from that connection, and prints it to the screen; then it closes the connection.

Server

```
import java.io.IOException;
import java.io.InputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class HelloServer {

    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(5556); // Choose any available port

            System.out.println("Server is running and waiting for a connection...");

            Socket clientSocket = serverSocket.accept(); // Wait for a client to connect
            System.out.println("Client connected: " + clientSocket);

            // Read data from the client
```

```
            InputStream input = clientSocket.getInputStream();
            byte[] buffer = new byte[1024];
            int bytesRead = input.read(buffer);

            if (bytesRead != -1) {
                String receivedData = new String(buffer, 0, bytesRead);
                System.out.println("Received data from client: " + receivedData);
            }

            // Close the connection
            clientSocket.close();
            serverSocket.close();

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

Client

```
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;

public class HelloClient {

    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost", 5556); // Connect to the server

            // Send data to the server
            String sendData = "hello_client";
            OutputStream output = socket.getOutputStream();
            output.write(sendData.getBytes());

            // Close the connection
            socket.close();

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

3. To implement a CHAT application using TCP Socket communication for client and server. Both can

run in the same machine or different machines.

Or

6. Write a program to implement the TCP Client Server Chat Application using Sockets.

Server

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.HashSet;
import java.util.Set;

public class TcpChatServer {
   private static final int PORT = 5550;
   private static Set<OutputStream> clients = new HashSet<>();

   public static void main(String[] args) {
      try {
         ServerSocket serverSocket = new ServerSocket(PORT);

         System.out.println("TCP Chat Server is running and waiting for clients...");

         while (true) {
            Socket clientSocket = serverSocket.accept();
            System.out.println("New client connected: " + clientSocket);

            OutputStream clientOutput = clientSocket.getOutputStream();
            clients.add(clientOutput);

            // Create a new thread to handle each client
            new Thread(() -> handleClient(clientSocket)).start();
         }
      } catch (IOException e) {
         e.printStackTrace();
      }
   }

   private static void handleClient(Socket clientSocket) {
      try {
         InputStream clientInput = clientSocket.getInputStream();
         byte[] buffer = new byte[1024];
```

```
            while (true) {
                int bytesRead = clientInput.read(buffer);
                if (bytesRead == -1) {
                    break; // Client disconnected
                }

                String message = new String(buffer, 0, bytesRead);
                System.out.println("Received message from client: " + message);

                // Broadcast the message to all connected clients
                broadcast(message);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                clientSocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    private static void broadcast(String message) {
        for (OutputStream clientOutput : clients) {
            try {
                clientOutput.write(message.getBytes());
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Client

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import java.util.Scanner;

public class TcpChatClient {
    private static final String SERVER_IP = "localhost";
    private static final int SERVER_PORT = 5550;

    public static void main(String[] args) {
        try {
            Socket socket = new Socket(SERVER_IP, SERVER_PORT);
```

```
        System.out.println("Connected to TCP Chat Server");

        OutputStream output = socket.getOutputStream();
        InputStream input = socket.getInputStream();

        new Thread(() -> {
            try {
                while (true) {
                    byte[] buffer = new byte[1024];
                    int bytesRead = input.read(buffer);

                    if (bytesRead == -1) {
                        System.out.println("Server has disconnected.");
                        System.exit(0);
                    }

                    String message = new String(buffer, 0, bytesRead);
                    System.out.println("Received message: " + message);
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }).start();

        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.print("Enter a message: ");
            String message = scanner.nextLine();

            output.write(message.getBytes());
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

4 Write program to implement DNS using UDP Sockets by getting any frame size based on the user request and send the frames to server..

Or

8. Write the program for Simulation of DNS using UDP Socket by getting the frame size from theuser and create the frame sized user request and send the frame to server from the client.

Server

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class DnsServer {
    private static final int PORT = 5551;

    public static void main(String[] args) {
        try {
            DatagramSocket socket = new DatagramSocket(PORT);

            System.out.println("DNS Server is running...");

            while (true) {
                byte[] receiveBuffer = new byte[1024];

                // Receive DNS request from the client
                DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
                socket.receive(receivePacket);

                // Extract the domain name from the request
                String domainName = new String(receivePacket.getData(), 0, receivePacket.getLength());
                System.out.println("Received DNS request for domain: " + domainName);

                // Perform DNS resolution (In this example, a simple mapping is used)
                String ipAddress = resolveIpAddress(domainName);

                // Send the IP address back to the client
                byte[] sendBuffer = ipAddress.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length,
receivePacket.getAddress(), receivePacket.getPort());
                socket.send(sendPacket);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static String resolveIpAddress(String domainName) {
        // In a real DNS server, this method would perform actual DNS resolution.
        // For simplicity, we use a simple mapping here.
        switch (domainName.toLowerCase()) {
            case "example.com":
                return "192.168.1.1";
            case "example.org":
                return "192.168.1.2";
            default:
                return "Unknown IP Address";
```

```
    }
  }
}
```

Client

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;

public class DnsClient {
  private static final String SERVER_IP = "localhost";
  private static final int SERVER_PORT = 5551;

  public static void main(String[] args) {
    try {
      Scanner scanner = new Scanner(System.in);

      while (true) {
        System.out.print("Enter a domain name (or 'exit' to quit): ");
        String domainName = scanner.nextLine();

        if (domainName.equalsIgnoreCase("exit")) {
          break;
        }

        // Send DNS request to the server
        DatagramSocket socket = new DatagramSocket();
        InetAddress serverAddress = InetAddress.getByName(SERVER_IP);
        byte[] sendBuffer = domainName.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length,
serverAddress, SERVER_PORT);
        socket.send(sendPacket);

        // Receive the IP address from the server
        byte[] receiveBuffer = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
        socket.receive(receivePacket);

        // Extract the IP address from the response
        String ipAddress = new String(receivePacket.getData(), 0, receivePacket.getLength());
        System.out.println("Resolved IP address: " + ipAddress);

        // Close the socket
        socket.close();
      }
    } catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }
}
```

Output

1. **Server Output:**

```
DNS Server is running...

Received DNS request for domain: example.com
```

Client Output:

```
Enter a domain name (or 'exit' to quit): example.com

Resolved IP address: 192.168.1.1
```

5. Write a HTTP web client program to download any web page using TCP sockets.

Server

```
import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.net.Socket;


public class HttpClient {


    public static void main(String[] args) {

        // Replace this URL with the desired web page URL

        String url = "http://www.example.com";


        try {

            // Extract host and path from the URL

            String host = getHost(url);

            String path = getPath(url);


            // Establish a TCP connection to the web server
```

```
        Socket socket = new Socket(host, 80); // Assuming HTTP, so using port 80

        // Send HTTP GET request to the server
        OutputStream out = socket.getOutputStream();
        out.write(("GET " + path + " HTTP/1.1\r\n").getBytes());
        out.write(("Host: " + host + "\r\n").getBytes());
        out.write("Connection: close\r\n".getBytes()); // Close the connection after the response
        out.write("\r\n".getBytes());
        out.flush();

        // Read and print the response from the server
        BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        String line;
        while ((line = in.readLine()) != null) {
            System.out.println(line);
        }

        // Close the connection
        socket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private static String getHost(String url) {
    // Extract host from the URL
    int start = url.indexOf("://") + 3;
    int end = url.indexOf("/", start);
    return url.substring(start, end);
}
```

```
    private static String getPath(String url) {
        // Extract path from the URL
        int start = url.indexOf("/", url.indexOf("://") + 3);
        return url.substring(start);
    }
}
```

7. Write a programs hello_server for TCP (The client connects to the server, sends the string "Hello, world!", then closes the connection)

Server

```
import java.io.IOException;
import java.io.InputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class HelloServer {

    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(5555); // Choose any available port

            System.out.println("Hello Server is running and waiting for a connection...");

            // Wait for a client to connect
            Socket clientSocket = serverSocket.accept();
            System.out.println("Client connected: " + clientSocket);

            // Receive data from the client
            InputStream input = clientSocket.getInputStream();
            byte[] buffer = new byte[1024];
            int bytesRead = input.read(buffer);
```

```
        if (bytesRead != -1) {
            String receivedData = new String(buffer, 0, bytesRead);
            System.out.println("Received from client: " + receivedData);
        }

        // Close the connection
        clientSocket.close();
        serverSocket.close();

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

Client

```
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;

public class HelloClient {

  public static void main(String[] args) {
    try {
      Socket socket = new Socket("localhost", 5555); // Connect to the server

      // Send data to the server
      String sendData = "Hello, world!";
      OutputStream output = socket.getOutputStream();
      output.write(sendData.getBytes());
```

```
            // Close the connection
            socket.close();

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