CRC

```
public class CRC {
    public static void main(String[] args) {
        String data = "11010011101100";
        String key = "1011";
        String crc = generateCRC(data, key);
        System.out.println("CRC Code: " + crc);
        System.out.println("Transmitted Frame: " + data + crc);
    }

    public static String generateCRC(String data, String key) {
        int keyLen = key.length();
        String appendedData = data + "0".repeat(keyLen - 1);
        char[] dataArr = appendedData.toCharArray();
        char[] keyArr = key.toCharArray();

        for (int i = 0; i <= appendedData.length() - keyLen; i++) {
            if (dataArr[i] == '1') {
                for (int j = 0; j < keyLen; j++) {
                    dataArr[i + j] = (dataArr[i + j] == keyArr[j]) ? '0' : '1';
                }
            }
        }

        return new String(dataArr).substring(dataArr.length - (keyLen - 1));
    }
}
```

STOPWAIT AQR

```
public class StopWaitARQ {
    public static void main(String[] args) {
        String[] frames = { "Frame1", "Frame2", "Frame3" };
        boolean ack;
        for (String frame : frames) {
            System.out.println("Sending: " + frame);
            ack = true; // Simulate ACK received
            if (ack) {
                System.out.println("ACK received for " + frame);
            } else {
                System.out.println("Resending " + frame);
            }
        }
    }
}
```

SocketProgrming_client

```
import java.net.*;
import java.io.*;

public class SocketProgramming_Client {
    public static void main(String[] args) throws IOException {
        Socket socket = new Socket("127.0.0.1", 5000);
        DataOutputStream out = new DataOutputStream(socket.getOutputStream());
        out.writeUTF("Hello Server");
        out.flush();
        out.close();
        socket.close();
    }
}
```

Server

```
import java.net.*;
import java.io.*;
public class SocketProgramming_Server {
    public static void main(String[] args) throws IOException {
        ServerSocket server = new ServerSocket(5000);
        System.out.println("Server is waiting for client...");
        Socket socket = server.accept();
        DataInputStream in = new DataInputStream(socket.getInputStream());
        System.out.println("Client says: " + in.readUTF());
        in.close();
        socket.close();
```

```
        server.close();
    }
}
```

TCP Day Time Client

```
import java.net.*;
import java.io.*;
public class TCPDayTimeClient {
    public static void main(String[] args) throws IOException {
        Socket socket = new Socket("127.0.0.1", 8000);
        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        System.out.println(in.readLine());
        in.close();
        socket.close();
    }
}
```

TCP Day Time Sever

```
import java.net.*;
import java.io.*;
import java.util.Date;
public class TCPDayTimeServer {
    public static void main(String[] args) throws IOException {
        ServerSocket server = new ServerSocket(8000);
        System.out.println("TCP Daytime Server waiting...");
        Socket socket = server.accept();
        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
        out.println("Server Date & Time: " + new Date().toString());
        out.close();
        socket.close();
```

```
        server.close();
    }
}
```

UDP Day Time Client

```
import java.net.*;
public class UDPDayTimeClient {
    public static void main(String[] args) throws Exception {
        DatagramSocket client = new DatagramSocket();
        InetAddress serverIP = InetAddress.getByName("localhost");
        byte[] request = "Send Date".getBytes();
        DatagramPacket packet = new DatagramPacket(request, request.length, serverIP, 9999);
        client.send(packet);
        byte[] responseBuf = new byte[1024];
        DatagramPacket response = new DatagramPacket(responseBuf, responseBuf.length);
        client.receive(response);
        System.out.println("Current Date & Time: " + new String(response.getData(), 0,
                response.getLength()));
        client.close();
    }
}
```

UDP Day Time Server

```
import java.net.*;
import java.util.Date;
public class UDPDayTimeServer {
    public static void main(String[] args) throws Exception {
        DatagramSocket server = new DatagramSocket(9999);
        byte[] buffer = new byte[1024];
```

```
        while (true) {
            DatagramPacket request = new DatagramPacket(buffer, buffer.length);
            server.receive(request);
            String date = new Date().toString();
            byte[] responseData = date.getBytes();
            DatagramPacket response = new DatagramPacket(responseData,
                responseData.length,
                request.getAddress(), request.getPort());
            server.send(response);
        }
    }
}
```

Command	Description / Purpose	Example Command (Windows)	Example Command (Linux)
ping	Tests connectivity between the local host and a remote host using ICMP packets. It checks if a destination is reachable.	ping www.google.com	ping www.google.com
ipconfig	Displays basic IP configuration details like IP address, subnet mask, and default gateway.	ipconfig	—
ipconfig /all	Displays detailed IP configuration including MAC address, DHCP, and DNS information.	ipconfig /all	—
ifconfig	Displays or configures network interfaces (used in Linux or older systems).	—	ifconfig -a
tracert / traceroute	Displays the route packets take from source to destination, showing each hop (network path).	tracert www.google.com	traceroute www.google.com
netstat	Displays active network connections, listening ports, and protocol statistics. Helps monitor network activity.	netstat -ano	netstat -tunlp

Command	Description / Purpose	Example Command (Windows)	Example Command (Linux)
nslookup	Queries the DNS server to resolve domain names to IP addresses (or vice versa).	nslookup www.google.com	nslookup www.google.com

ping

Description:
Used to test connectivity between your computer (local host) and another device or website on a network. It sends ICMP packets and shows whether the destination is reachable and how long it takes.
Example:
```
ping www.google.com
```

---

ipconfig

Description:
Displays the basic IP configuration details of your computer, such as IP address, subnet mask, and default gateway.
Example:
```
ipconfig
```

---

ipconfig /all

Description:
Shows detailed network configuration information including MAC address, DNS servers, DHCP status, and more.
Example:
```
ipconfig /all
```

---

tracert

Description:
Displays the route (path) packets take from your computer to a destination host, showing each hop along the way. It helps identify where network delays occur.
Example:
```
tracert www.google.com
```

---

netstat

Description:
Shows active network connections, listening ports, and protocol statistics. Useful for monitoring network activity and troubleshooting connections.
Example:
```
netstat -ano
```

---

nslookup

Description:
Used to query DNS servers and find the IP address associated with a domain name (or vice versa). Helps check if DNS resolution is working properly.
Example:
```
nslookup www.google.com
```

---

getmac

Description:
Displays the physical (MAC) address of your network adapters. Useful for identifying your system's unique hardware address.
Example:
```
getmac
```