1. Check Computer Connectivity Using the ping Command

```
import java.io.*;

public class PT {
    public static void main(String[] args) {
        String cmd = "ping -n 4 google.com ";

        try {
            Process p = Runtime.getRuntime().exec(cmd);
            BufferedReader r = new BufferedReader(new
InputStreamReader(p.getInputStream()));
            String line; boolean reachable = false;
            while ((line = r.readLine()) != null) {
                System.out.println(line);
                if (line.contains("Reply from") || line.contains("bytes
from")) reachable = true;
            }
            System.out.println("\nResult: Host is " + (reachable ?
"reachable." : "unreachable."));
        } catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
        }
    }
}
```

Output:

```
Pinging google.com [2404:6800:4007:83e::200e] with 32 bytes of data:
Reply from 2404:6800:4007:83e::200e: time=280ms
Reply from 2404:6800:4007:83e::200e: time=491ms
Reply from 2404:6800:4007:83e::200e: time=1224ms
Reply from 2404:6800:4007:83e::200e: time=782ms

Ping statistics for 2404:6800:4007:83e::200e:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 280ms, Maximum = 1224ms, Average = 694ms

Result: Host is reachable.
PS D:\Projects\Practical\CN>
```

2. Trace Packet Routes Using the traceroute Command

```
import java.io.*;
public class TracerouteSimulator {
    public static void main(String[] a) throws Exception {
        String host = "google.com";
        System.out.println("Tracing route to " + host + "...\n");
        Process p = new ProcessBuilder("tracert", host).start();
        BufferedReader r = new BufferedReader(new
InputStreamReader(p.getInputStream()));
        String l; while ((l = r.readLine()) != null) System.out.println(l);
        System.out.println("\nDone (exit " + p.waitFor() + ")");
    }
}
```

Output:

```
Tracing route to google.com [2404:6800:4007:83e::200e]
over a maximum of 30 hops:

  1     3 ms     2 ms     2 ms  2409:40f4:39:1c28::17
  2     *        *        *     Request timed out.
  3   611 ms   100 ms   157 ms  2405:200:5218:21:3925::1
  4   227 ms   224 ms   236 ms  2405:200:88c:1513:62::6
  5     *        *        *     Request timed out.
  6   276 ms   189 ms   399 ms  2405:200:801:900::1624
  7     *        *        *     Request timed out.
  8     *        *        *     Request timed out.
  9     *        *        *     Request timed out.
 10   788 ms   501 ms   247 ms  2404:6800:8201:200::1
 11   342 ms   222 ms  1333 ms  2001:4860:0:1::856
 12  1022 ms  1249 ms   784 ms  2001:4860:0:1::5595
 13   219 ms   242 ms   250 ms  lcmaaa-bc-in-x0e.1e100.net [2404:6800:4007:83e::200e]

Trace complete.

Done (exit 0)
```

3. Mounting a Remote Computer's Volume using net use Command

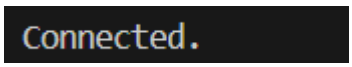
Server code:

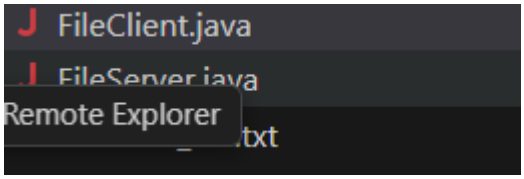
```
import java.io.*; import java.net.*;
public class FileServer {
    public static void main(String[] a) throws Exception {
        ServerSocket ss = new ServerSocket(1234);
        System.out.println("Waiting...");
        Socket s = ss.accept();
        System.out.println("Client connected.");
        try (InputStream in = new FileInputStream("shared_file.txt");
             OutputStream out = s.getOutputStream()) {
            in.transferTo(out);
        }
        s.close(); ss.close();
        System.out.println("File sent.");
    }
}
```

Client code:

```
import java.io.*; import java.net.*;
public class FileClient {
    public static void main(String[] a) throws Exception {
        Socket s = new Socket("localhost", 1234);
        System.out.println("Connected.");
        try (InputStream in = s.getInputStream();
             OutputStream out = new FileOutputStream("received_file.txt")) {
            in.transferTo(out);
        }
        s.close();
        System.out.println("File received.");
    }
}
```

Output:





4. Send Messages Between Machines Using Socket Programming

```
 Server code:
import java.io.*; import java.net.*;
public class TCPServer {
    public static void main(String[] a) throws IOException {
        ServerSocket ss = new ServerSocket(1234);
        System.out.println("Waiting...");
        Socket s = ss.accept();
        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        PrintWriter out = new PrintWriter(s.getOutputStream(), true);
        String msg = in.readLine();
        System.out.println("Client: " + msg);
        out.println("Server reply: " + msg);
        s.close(); ss.close();
    }
}
```

```
Client code:

import java.io.*; import java.net.*;
public class TCPClient {
    public static void main(String[] a) throws IOException {
        Socket s = new Socket("localhost", 1234);
        PrintWriter out = new PrintWriter(s.getOutputStream(), true);
        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        System.out.print("Enter message: ");
        String msg = new BufferedReader(new
InputStreamReader(System.in)).readLine();
        out.println(msg);
        System.out.println(in.readLine());
        s.close();
    }
}
```

Output:

```
Enter message: Hii
Server reply: Hii
```

5. Simulate a Chatting Application using Socket in Java

Server code:

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

public class ChatServer {
    public static void main(String[] args) throws IOException {
        ServerSocket server = new ServerSocket(1234);
        System.out.println("Waiting for client...");
        Socket socket = server.accept();
        System.out.println("Client connected.");

        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
        Scanner sc = new Scanner(System.in);

        // Send messages
        new Thread(() -> {
            while (true) {
                System.out.print("Enter your message: ");
                String msg = sc.nextLine();
                out.println(msg);
                if (msg.equalsIgnoreCase("exit")) System.exit(0);
            }
        }).start();

        // Receive messages
        String msg;
        while ((msg = in.readLine()) != null) {
            System.out.println("\nClient: " + msg);
            if (msg.equalsIgnoreCase("exit")) System.exit(0);
        }
    }
}
```

Client code:

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

public class ChatClient {
    public static void main(String[] args) throws IOException {
        Socket socket = new Socket("localhost", 1234);
        System.out.println("Connected to server.");

        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
        Scanner sc = new Scanner(System.in);

        // Send messages
        new Thread(() -> {
            while (true) {
                System.out.print("Enter your message: ");
                String msg = sc.nextLine();
                out.println(msg);
                if (msg.equalsIgnoreCase("exit")) System.exit(0);
            }
        }).start();

        // Receive messages
        String msg;
        while ((msg = in.readLine()) != null) {
            System.out.println("\nServer: " + msg);
            if (msg.equalsIgnoreCase("exit")) System.exit(0);
        }
    }
}
```

Output:

```
Waiting for client...
Client connected.
Enter your message:
Client: Hii
```

6. Implementation of File Transfer Protocol (FTP) in Java

FTP Server code

```
import java.io.*; import java.net.*;
public class FTPServer {
    public static void main(String[] args) throws IOException {
        ServerSocket server = new ServerSocket(5000);
        System.out.println("FTP Server started on port 5000...");
        while(true){
            Socket s = server.accept();
            System.out.println("Client connected: " + s.getInetAddress());
            DataInputStream dis = new DataInputStream(s.getInputStream());
            DataOutputStream dos = new DataOutputStream(s.getOutputStream());
            String cmd = dis.readUTF();
            if(cmd.equalsIgnoreCase("UPLOAD")){
                String f = dis.readUTF();
                try(FileOutputStream fos=new FileOutputStream(f)){
                    byte[] buf=new byte[4096]; int n;
                    while((n=dis.read(buf))!=-1) fos.write(buf,0,n);
                }
                System.out.println("File received: "+f);
            } else if(cmd.equalsIgnoreCase("DOWNLOAD")){
                String f = dis.readUTF(); File file = new File(f);
                if(file.exists()){
                    dos.writeUTF("FILE_EXISTS");
                    try(FileInputStream fis=new FileInputStream(file)){
                        byte[] buf=new byte[4096]; int n;
                        while((n=fis.read(buf))!=-1) dos.write(buf,0,n);
                    }
                    System.out.println("File sent: "+f);
                } else dos.writeUTF("FILE_NOT_FOUND");
            }
            s.close();
        }
    }
}
```

FTP Client code

```
import java.io.*; import java.net.*;
public class FTPClient {
    public static void main(String[] args) throws IOException {
        Socket s = new Socket("localhost",5000);
        DataInputStream dis = new DataInputStream(s.getInputStream());
        DataOutputStream dos = new DataOutputStream(s.getOutputStream());
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        System.out.print("Enter command (UPLOAD/DOWNLOAD): "); String
cmd=br.readLine();
        dos.writeUTF(cmd);
        if(cmd.equalsIgnoreCase("UPLOAD")){
            System.out.print("Enter file path: "); String path=br.readLine();
            File file=new File(path);
            if(file.exists()){
                dos.writeUTF(file.getName());
                try(FileInputStream fis=new FileInputStream(file)){
                    byte[] buf=new byte[4096]; int n;
                    while((n=fis.read(buf))!=-1) dos.write(buf,0,n);
                }
                System.out.println("File uploaded!");
            } else System.out.println("File not found!");
        } else if(cmd.equalsIgnoreCase("DOWNLOAD")){
            System.out.print("Enter filename: "); String f=br.readLine();
            dos.writeUTF(f);
            if(dis.readUTF().equals("FILE_EXISTS")){
                try(FileOutputStream fos=new
FileOutputStream("downloaded_"+f)){
                    byte[] buf=new byte[4096]; int n;
                    while((n=dis.read(buf))!=-1) fos.write(buf,0,n);
                }
                System.out.println("File downloaded!");
            } else System.out.println("File not found on server!");
        }
        s.close();
    }
}
```

```
Enter command (UPLOAD/DOWNLOAD): UPLOAD
Enter file path: D:\Projects\Practical\CN\CN.pdf
File uploaded!
```

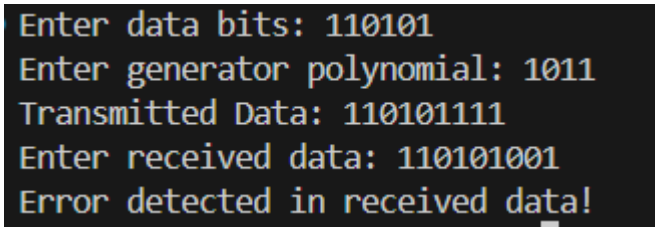
```
FTP Server started on port 5000...
Client connected: /127.0.0.1
File received: CN.pdf
```

7. Implementation of CRC and Hamming Code for Error Handling in Java

```
import java.util.Scanner;

public class CRC {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Input data bits and generator polynomial
        System.out.print("Enter data bits: ");
        String data = sc.next();
        System.out.print("Enter generator polynomial: ");
        String generator = sc.next();

        // Calculate CRC and display transmitted data
        String transmittedData = calculateCRC(data, generator);
        System.out.println("Transmitted Data: " + transmittedData);

        // Input received data and check for errors
        System.out.print("Enter received data: ");
        String receivedData = sc.next();
        if (checkError(receivedData, generator)) {
            System.out.println("No error detected.");
        } else {
            System.out.println("Error detected in received data!");
        }

        sc.close();
    }

    // Method to calculate CRC and append to data
    public static String calculateCRC(String data, String generator) {
        int genLength = generator.length();
        StringBuilder paddedData = new StringBuilder(data);

        // Append zeros to match generator length
        for (int i = 0; i < genLength - 1; i++) {
            paddedData.append('0');
        }

        // Perform XOR division
        for (int i = 0; i <= paddedData.length() - genLength;) {
            for (int j = 0; j < genLength; j++) {
                paddedData.setCharAt(i + j, paddedData.charAt(i + j) ==
generator.charAt(j) ? '0' : '1');
            }
```

```
            while (i < paddedData.length() && paddedData.charAt(i) == '0')
{
                i++;
            }
        }

        // Extract remainder and append to original data
        String crc = paddedData.substring(paddedData.length() - genLength +
1);
        return data + crc;
    }

    // Method to check for errors in received data
    public static boolean checkError(String receivedData, String generator)
{
        int genLength = generator.length();
        StringBuilder temp = new StringBuilder(receivedData);

        // Perform XOR division
        for (int i = 0; i <= temp.length() - genLength;) {
            for (int j = 0; j < genLength; j++) {
                temp.setCharAt(i + j, temp.charAt(i + j) ==
generator.charAt(j) ? '0' : '1');
            }
            while (i < temp.length() && temp.charAt(i) == '0') {
                i++;
            }
        }

        // Check if remainder is all zeros
        for (int i = temp.length() - genLength + 1; i < temp.length(); i++)
{
            if (temp.charAt(i) != '0') {
                return false;
            }
        }
        return true;
    }
}
```

Output:


Enter data bits: 110101
Enter generator polynomial: 1011
Transmitted Data: 110101111
Enter received data: 110101001
Error detected in received data!

8. Hamming Code (Error Correction)

```java
import java.util.Scanner;

public class HammingCode {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter 4-bit data: ");
        String data = sc.next();
        int[] code = encodeHamming(data);
        System.out.print("Transmitted Code: ");
        for (int b : code) System.out.print(b);

        System.out.print("\nEnter received 7-bit code: ");
        String r = sc.next();
        int err = detectError(r);
        System.out.println(err == 0 ? "No error detected." : "Error at
position: " + err);
        sc.close();
    }

    static int[] encodeHamming(String d) {
        int[] c = new int[7];
        c[2]=d.charAt(0)-'0'; c[4]=d.charAt(1)-'0'; c[5]=d.charAt(2)-'0';
c[6]=d.charAt(3)-'0';
        c[0]=c[2]^c[4]^c[6]; c[1]=c[2]^c[5]^c[6]; c[3]=c[4]^c[5]^c[6];
        return c;
    }

    static int detectError(String r) {
        int p1=r.charAt(0)-'0'^r.charAt(2)-'0'^r.charAt(4)-'0'^r.charAt(6)-
'0';
        int p2=r.charAt(1)-'0'^r.charAt(2)-'0'^r.charAt(5)-'0'^r.charAt(6)-
'0';
        int p4=r.charAt(3)-'0'^r.charAt(4)-'0'^r.charAt(5)-'0'^r.charAt(6)-
'0';
        return p1 + p2*2 + p4*4;
    }
}
```

```
Enter 4-bit data: 1011
Transmitted Code: 0110011
Enter received 7-bit code: 0100011
Error at position: 3
```

9. Client-Server Program using TCP and UDP Sockets in Java

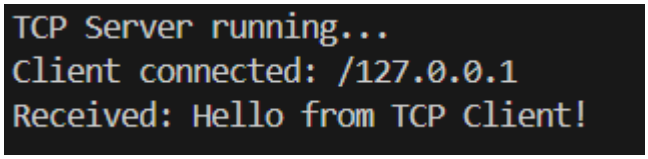
Server code:

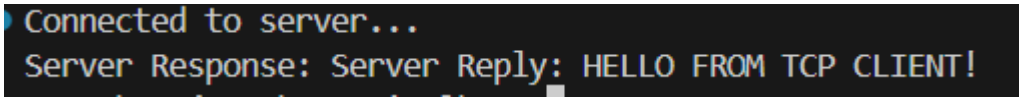
```
import java.io.*; import java.net.*;

public class TCPServer {
    public static void main(String[] args) throws IOException {
        ServerSocket server = new ServerSocket(6789);
        System.out.println("TCP Server running...");
        while (true) {
            Socket s = server.accept();
            System.out.println("Client connected: " + s.getInetAddress());
            BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
            PrintWriter out = new PrintWriter(s.getOutputStream(), true);
            String msg = in.readLine();
            System.out.println("Received: " + msg);
            out.println("Server Reply: " + msg.toUpperCase());
            s.close();
        }
    }
}
```

Client code:

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

public class TCPClient {
    public static void main(String[] args) throws IOException {
        Socket s = new Socket("localhost", 6789);
        System.out.println("Connected to server...");
        PrintWriter out = new PrintWriter(s.getOutputStream(), true);
        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        out.println("Hello from TCP Client!");
        System.out.println("Server Response: " + in.readLine());
        s.close();
    }
}
```

Output:


TCP Server running...
Client connected: /127.0.0.1
Received: Hello from TCP Client!


Connected to server...
Server Response: Server Reply: HELLO FROM TCP CLIENT!

10. UDP Socket Communication (Connectionless)

```
UDP Server:
import java.net.*;

public class UDPServer {
    public static void main(String[] args) throws Exception {
        DatagramSocket server = new DatagramSocket(9876);
        System.out.println("UDP Server running...");
        byte[] buf = new byte[1024];

        while (true) {
            DatagramPacket recv = new DatagramPacket(buf, buf.length);
            server.receive(recv);
            String msg = new String(recv.getData(), 0, recv.getLength());
            System.out.println("Received: " + msg);

            String reply = "Server Reply: " + msg.toUpperCase();
            byte[] sendData = reply.getBytes();
            DatagramPacket send = new DatagramPacket(sendData,
sendData.length,
                    recv.getAddress(), recv.getPort());
            server.send(send);
        }
    }
}
```

```
UDP Client:
import java.net.*;
import java.util.Scanner;

public class UDPClient {
    public static void main(String[] args) throws Exception {
        DatagramSocket client = new DatagramSocket();
        InetAddress serverIP = InetAddress.getByName("localhost");
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter message: ");
        String msg = sc.nextLine();
        byte[] sendData = msg.getBytes();
        client.send(new DatagramPacket(sendData, sendData.length, serverIP,
9876));

        byte[] recvData = new byte[1024];
        DatagramPacket recv = new DatagramPacket(recvData, recvData.length);
        client.receive(recv);
```

```
        System.out.println("Server Response: " + new String(recv.getData(), 0,
recv.getLength()));

        client.close();
        sc.close();
    }
}
```

Output:

```
UDP Server running...
Received: Namaste
```

```
Enter message: Namaste
Server Response: Server Reply: NAMASTE
```

11. ARP to RARP

```
import java.util.*;
public class ARP_RARP_Simulation{
    static Map<String,String> a=new HashMap<>(),r=new HashMap<>();
    public static void main(String[] x){
        a.put("192.168.1.1","00:1A:2B:3C:4D:5E");
        a.put("192.168.1.2","00:2B:3C:4D:5E:6F");
        r.put("00:1A:2B:3C:4D:5E","192.168.1.1");
        r.put("00:2B:3C:4D:5E:6F","192.168.1.2");
        Scanner s=new Scanner(System.in);
        System.out.print("1.ARP 2.RARP> ");
        int c=s.nextInt();
        s.nextLine();
        System.out.println(c==1?"MAC:"+a.getOrDefault(s.nextLine(),"Not
Found"):
                           c==2?"IP:"+r.getOrDefault(s.nextLine(),"Not
Found"):"Invalid!");
    }
}
```

Output:

```
1.ARP 2.RARP> 1
192.168.1.1
MAC:00:1A:2B:3C:4D:5E
```

12. Socket Programming: Echo, Ping, and Talk Commands in Java

Server code:

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

public class EchoServer {
    public static void main(String[] args) throws IOException {
        ServerSocket server = new ServerSocket(5000);
        System.out.println("Echo Server started...");
        Socket s = server.accept();
        System.out.println("Client connected.");

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

        String msg;
        while ((msg = in.readLine()) != null) {
            System.out.println("Received: " + msg);
            out.println("Server echoes: " + msg);
        }

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

Client code:

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

public class EchoClient {
    public static void main(String[] args) throws IOException {
        Socket s = new Socket("localhost", 5000);
        System.out.println("Connected to Echo Server.");

        PrintWriter out = new PrintWriter(s.getOutputStream(), true);
        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter message: ");
        String msg = sc.nextLine();
        out.println(msg);
        System.out.println("Server response: " + in.readLine());
```

```
        s.close();
        sc.close();
    }
}
```

Output:

```
Echo Server started...
Client connected.
Received: Hi
```

```
Connected to Echo Server.
Enter message: Hi
Server response: Server echoes: Hi
```

13. PING

Server code:

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

public class PingServer {
    public static void main(String[] args) throws IOException {
        ServerSocket server = new ServerSocket(6000);
        System.out.println("Ping Server started...");
        Socket s = server.accept();
        System.out.println("Client connected.");

        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        PrintWriter out = new PrintWriter(s.getOutputStream(), true);
        String msg;
        while ((msg = in.readLine()) != null) {
            if (msg.equalsIgnoreCase("ping")) {
                out.println("pong");
                System.out.println("Ping received. Sent pong.");
            }
        }

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

Client code:

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

public class PingClient {
    public static void main(String[] args) throws IOException {
        Socket s = new Socket("localhost", 6000);
        System.out.println("Connected to Ping Server.");

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

        long start = System.currentTimeMillis();
        out.println("ping");
        if (in.readLine().equals("pong")) {
```

```
            long rtt = System.currentTimeMillis() - start;
            System.out.println("Ping successful. RTT: " + rtt + " ms");
        }

        s.close();
    }
}
```

Output:

```
Ping Server started...
Client connected.
Ping received. Sent pong.
```

```
Connected to Ping Server.
Ping successful. RTT: 3 ms
```

14. TALK

Server code:

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

public class TalkServer {
    public static void main(String[] args) throws IOException {
        ServerSocket server = new ServerSocket(7000);
        System.out.println("Talk Server started...");
        Socket s = server.accept();
        System.out.println("Client connected.");

        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        PrintWriter out = new PrintWriter(s.getOutputStream(), true);
        BufferedReader serverInput = new BufferedReader(new
InputStreamReader(System.in));

        String clientMsg, serverMsg;
        while (true) {
            clientMsg = in.readLine();
            System.out.println("Client says: " + clientMsg);
            System.out.print("Server reply: ");
            serverMsg = serverInput.readLine();
            out.println(serverMsg);
        }
    }
}
```

Client code:

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

public class TalkClient {
    public static void main(String[] args) throws IOException {
        Socket s = new Socket("localhost", 7000);
        System.out.println("Connected to Talk Server.");

        PrintWriter out = new PrintWriter(s.getOutputStream(), true);
        BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
        Scanner sc = new Scanner(System.in);
```

```
        String userMsg, serverResp;
        while (true) {
            System.out.print("Client: ");
            userMsg = sc.nextLine();
            out.println(userMsg);
            serverResp = in.readLine();
            System.out.println("Server says: " + serverResp);
        }
    }
}
```

Output:

```
Talk Server started...
Client connected.
Client says: Hii
Server reply: Hello
```

```
Connected to Talk Server.
Client: Hii
Server says: Hello
Client:
```