Project 2 Task 0

Project2Task0Client

```
/**
 * This program implements a UDP client.
 * @author Candice Chiang
 * Andrew id: wantienc
 */

// Import the necessary packages for UDP
import java.net.*;
import java.io.*;
import java.util.Scanner;

public class EchoClientUDP{
    /**
     * Implement a UDP client.
     * @param args Array of strings giving message contents and server hostname
     */
    public static void main(String args[]){
        // Announce the client starts running
        System.out.println("The UDP client is running.");
        // Get the server side port number from user
        // For this project, use 6789
        Scanner getDestPort = new Scanner(System.in);
        System.out.println("Insert the server side port number: ");
        int serverPort = getDestPort.nextInt();
        // Declare a Datagram (UDP style) socket
        DatagramSocket aSocket = null;
        try {
            // Collect the IP address
            InetAddress aHost = InetAddress.getByName("localhost");
            // Create the socket
            aSocket = new DatagramSocket();
            String nextLine;
            // Initialize a BufferedReader to read input from the console
            BufferedReader typed = new BufferedReader(new InputStreamReader(System.in));
            // Read lines of input
            while ((nextLine = typed.readLine()) != null) {
                // Convert the line into byte array
                byte [] m = nextLine.getBytes();
                /*
                Build the packet holding the byte message from the console, length of the
message,
                destination address, and the destination port number.
                 */
                DatagramPacket request = new DatagramPacket(m,  m.length, aHost,
serverPort);
                // Send the Datagram request on the socket
                aSocket.send(request);
                // Prepare buffer for the reply
```

```
                byte[] buffer = new byte[1000];
                // Create a Datagram for the reply
                DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
                // Wait and receive the reply
                aSocket.receive(reply);
                String replyStr = new String(reply.getData()).substring(0,
reply.getLength());
                // Show the result to the client
                System.out.println("Reply from server: " + replyStr);

                // Quit the client when user requests halt! and get response halt! by
server
                if(replyStr.equals("halt!")) {
                    System.out.println("UDP Client side quitting");
                    break;
                }

            }
        // Handle socket exceptions
        }catch (SocketException e) {System.out.println("Socket Exception: " +
e.getMessage());
            // Handle general IO exceptions
        }catch (IOException e){System.out.println("IO Exception: " + e.getMessage());
            // Close the socket if not null
        }finally {if(aSocket != null) aSocket.close();}
    }
}
```

Project2Task0Server

```
/**
 * This program implements a UDP server.
 * @author Candice Chiang
 * Andrew id: wantienc
 */
// Import the necessary packages for UDP
import java.net.*;
import java.io.*;
import java.util.Scanner;

public class EchoServerUDP{
    /**
     * Implement a UDP server.
     * @param args Array of strings from the console
     */
    public static void main(String args[]){
        // Announce the server starts running
        System.out.println("The UDP server is running.");
        // Get the port number this server to listen on from user
        // For this project, use 6789
        Scanner getPort = new Scanner(System.in);
        System.out.println("Insert the server side port number: ");
        int serverPort = getPort.nextInt();
        // Declare a Datagram (UDP style) socket
        DatagramSocket aSocket = null;
        // Prepare buffer
        byte[] buffer = new byte[1000];
        try{
            // Create a new DatagramSocket and bind it to port number from user input
            aSocket = new DatagramSocket(serverPort);
            // Create a new DatagramPacket for receiving requests
            DatagramPacket request = new DatagramPacket(buffer, buffer.length);
            // An infinite loop to wait for incoming datagrams
            while(true){
                // Receive a datagram
                aSocket.receive(request);
                /*
                Create a new DatagramPacket for sending replies
                with request's data, length, address, and port number.
                 */
                DatagramPacket reply = new DatagramPacket(request.getData(),
                        request.getLength(), request.getAddress(), request.getPort());
                // Convert the request data to a String
                String requestString = new String(request.getData()).substring(0,
request.getLength());

                // Print the request string
                System.out.println("Echoing: " + requestString);

                // Send a reply datagram back to the client
                aSocket.send(reply);
```

```
                // Quit the server when user requests halt!
                if(requestString.equals("halt!")){
                    System.out.println("UDP Server side quitting");
                    break;
                }
            }
            // Handle socket exceptions
        }catch (SocketException e){System.out.println("Socket: " + e.getMessage());
            // Handle general IO exceptions
        }catch (IOException e) {System.out.println("IO: " + e.getMessage());
            // Close the socket if not null
        }finally {if(aSocket != null) aSocket.close();}
    }
}
```

Project2Task0ClientConsole

```
Run:  EchoServerUDP  EchoClientUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/
The UDP client is running.
Insert the server side port number:
6789
Hello
Reply from server: Hello
ProjectTask0
Reply from server: ProjectTask0
20230220
Reply from server: 20230220
Heinz
Reply from server: Heinz
Distributed Systems
Reply from server: Distributed Systems
halt!
Reply from server: halt!
UDP Client side quitting

Process finished with exit code 0
```

Project2Task0ServerConsole

```
Run:  EchoServerUDP  EchoClientUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/
The UDP server is running.
Insert the server side port number:
6789
Echoing: Hello
Echoing: ProjectTask0
Echoing: 20230220
Echoing: Heinz
Echoing: Distributed Systems
Echoing: halt!
UDP Server side quitting

Process finished with exit code 0
```

Project 2 Task 1

EavesdropperUDP.java

```
/**
 * This program acts as a malicious player between server and client.
 * @author Candice Chiang
 * Andrew id: wantienc
 * Last Modified: Feb 20, 2023
 */
// Import the necessary packages for UDP
import java.net.*;
import java.io.*;
import java.util.Scanner;
public class EavesdropperUDP {
    /**
     * Implement an Eavesdropper.
     * @param args Array of strings from the console
     */
    public static void main(String args[]){
        // Announce the Eavesdropper starts running
        System.out.println("The Eavesdropper is running.");
        // Get the port number this server to listen on from user
        // For this project, use 6789
        Scanner getPort = new Scanner(System.in);
        System.out.println("Enter the port number to listen on: ");
        int listenPort = getPort.nextInt();
        // Get the port number to masquerade as from user
        // For this project, use 6798
        System.out.println("Enter the port number of the server to masquerade as: ");
        int masqueradPort = getPort.nextInt();
        // Declare a Datagram (UDP style) socket between client
        DatagramSocket aSocket = null;
        // Declare a Datagram (UDP style) socket between server
        DatagramSocket bSocket = null;
        // Prepare buffer
        byte[] buffer = new byte[1000];
        try{
            // Collect the IP address
            InetAddress aHost = InetAddress.getByName("localhost");
            // Create a new DatagramSocket and bind it to port number from user input
            aSocket = new DatagramSocket(listenPort);
            // Create the socket
            bSocket = new DatagramSocket();
            // Create a new DatagramPacket for receiving requests from client
            DatagramPacket trueRequest = new DatagramPacket(buffer, buffer.length);
            // An infinite loop to wait for incoming datagrams
            while(true){
                // Receive a datagram from client
                aSocket.receive(trueRequest);

                // Convert the request data to a String
```

```
                String requestString = new String(trueRequest.getData()).substring(0,
trueRequest.getLength());

                // Print the actual request string
                System.out.println("Message from Client: " + requestString);
                // Check if the message is general or requesting to quit
                if (!requestString.equals("halt!")) {
                    // Add ! to the actual message
                    byte[] m = (requestString + "!").getBytes();
                    String fakeRequestString = new String(m);
                    // Print the fake message to send to server
                    System.out.println("Send fake message to server: " +
fakeRequestString);
                    /*
                    Build the packet holding the byte message from the console, length of
the message,
                    destination address, and the destination port number.
                     */
                    DatagramPacket fakeRequest = new DatagramPacket(m, m.length, aHost,
masqueradPort);
                    // Send the Datagram request on the socket to server
                    bSocket.send(fakeRequest);
                } else {
                    /*
                    Build the packet holding the byte message from the console, length of
the message,
                    destination address, and the destination port number.
                     */
                    DatagramPacket request = new DatagramPacket(trueRequest.getData(),
trueRequest.getLength(),
                                                                aHost, masqueradPort);
                    System.out.println("Client requests quitting");
                    // Send the Datagram request on the socket to server
                    bSocket.send(request);
                }
                // Create a Datagram for the reply from server
                DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
                // Wait and receive the reply
                bSocket.receive(reply);
                String replyStr = new String(reply.getData()).substring(0,
reply.getLength());
                // Show the result from server
                System.out.println("Reply from server: " + replyStr);
                // Check if sever replies with halt!
                if (!requestString.equals("halt!")) {
                    // Remove ! from the reply
                    byte[] mock_m = replyStr.substring(0, replyStr.length() -
1).getBytes();
                    String fakeReplyString = new String(mock_m);
                    System.out.println("Reply the original message: " + fakeReplyString);
                    /*
                    Build the packet holding the byte message from the console, length of
the message,
                    destination address, and the destination port number.
```

```
                     */
                    DatagramPacket fakeReply = new DatagramPacket(mock_m, mock_m.length,
                                                        trueRequest.getAddress(),
trueRequest.getPort());
                    // Send a reply datagram back to the client
                    aSocket.send(fakeReply);
                } else {
                    /*
                    Build the packet holding the byte message to halt from the console,
length of the message,
                    destination address, and the destination port number.
                     */
                    DatagramPacket haltReply = new DatagramPacket(reply.getData(),
reply.getLength(),
                                                        trueRequest.getAddress(),
trueRequest.getPort());
                    System.out.println("Server quits");
                    // Send a reply datagram back to the client
                    aSocket.send(haltReply);
                }

            }
            // Handle socket exceptions
        }catch (SocketException e){System.out.println("Socket: " + e.getMessage());
            // Handle general IO exceptions
        }catch (IOException e) {System.out.println("IO: " + e.getMessage());
            // Close the socket if not null
        }finally {if(aSocket != null) aSocket.close();}
    }
}
```

Project2Task1ThreeConsoles

1. Client using port 6789 (correct server)
   a. Client console

```
Run:  EchoServerUDP  EavesdropperUDP  EchoClientUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/C
The UDP client is running.
Insert the server side port number:
6789
Hi
Reply from server: Hi
Hello
Reply from server: Hello
halt!
Reply from server: halt!
UDP Client side quitting

Process finished with exit code 0
```

   b. Server console

```
Run:  EchoServerUDP  EavesdropperUDP  EchoClientUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/
The UDP server is running.
Insert the server side port number:
6789
Echoing: Hi
Echoing: Hello
Echoing: halt!
UDP Server side quitting

Process finished with exit code 0
```

c. Eavesdropper console

```
Run:  EchoServerUDP  EavesdropperUDP  EchoClientUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Conte
The Eavesdropper is running.
Enter the port number to listen on:
6798
Enter the port number of the server to masquerade as:
6789
```

2. Client using port 6798 (malicious player)

a. Client console

```
Run:  EchoServerUDP  EavesdropperUDP  EchoClientUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/C
The UDP client is running.
Insert the server side port number:
6798
Hi
Reply from server: Hi
Hello
Reply from server: Hello
halt!
Reply from server: halt!
UDP Client side quitting

Process finished with exit code 0
```

b. Server console

```
Run:  EchoServerUDP  EavesdropperUDP  EchoClientUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/
The UDP server is running.
Insert the server side port number:
6789
Echoing: Hi!
Echoing: Hello!
Echoing: halt!
UDP Server side quitting

Process finished with exit code 0
```

c. Eavesdropper console

```
Run:  EchoServerUDP  EavesdropperUDP  EchoClientUDP
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Conte
The Eavesdropper is running.
Enter the port number to listen on:
6798
Enter the port number of the server to masquerade as:
6789
Message from Client: Hi
Send fake message to server: Hi!
Reply from server: Hi!
Reply the original message: Hi
Message from Client: Hello
Send fake message to server: Hello!
Reply from server: Hello!
Reply the original message: Hello
Message from Client: halt!
Client requests quitting
Reply from server: halt!
Server quits
```

Project 2 Task 2

Project2Task2Client

```
/**
 * This program implements a UDP client.
 * @author Candice Chiang
 * Andrew id: wantienc
 * Last Modified: Feb 20, 2023
 */

// Import the necessary packages for UDP
import java.net.*;
import java.io.*;
import java.nio.ByteBuffer;
import java.util.Scanner;

public class AddingClientUDP{
    static int serverPort;
    static InetAddress aHost;
    /**
     * Implement a UDP client.
     * @param args Array of strings giving message contents and server hostname
     */
    public static void main(String args[]){
        // Announce the client starts running
        System.out.println("The UDP client is running.");
        // Get the server side port number from user
        // For this project, use 6789
        Scanner getDestPort = new Scanner(System.in);
        System.out.println("Please enter server port: ");
        serverPort = getDestPort.nextInt();
        System.out.println();
        try {
            String nextLine;
            // Collect the IP address
            aHost = InetAddress.getByName("localhost");
            // Initialize a BufferedReader to read input from the console
            BufferedReader typed = new BufferedReader(new InputStreamReader(System.in));
            // Read lines of input
            while ((nextLine = typed.readLine()) != null) {
                // if the input matches integer format
                if (nextLine.matches("^[+-]*[0-9]+$")) {
                    // Convert the input into an integer
                    int num = Integer.parseInt(nextLine);
                    // Call add method to communicate with server and get the current sum
                    int replySum = add(num);
                    // Print result to console
                    System.out.println("The server returned " + replySum + ".");
                } else if (nextLine.equals("halt!")) { // if client requests to quit
                    System.out.println("Client side quitting.");
                    break;
```

```
                } else { // input other than integer or halt, continue to loop
                    System.out.println("Please enter an integer.");
                    continue;
                }
            }
            // Handle unknown host exceptions
        } catch (UnknownHostException e) {
            throw new RuntimeException(e);
            // Handle general IO exceptions
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

    }

    public static int add (int i) {
        // int sum to record the reply sum
        int sum;
        // Convert i into byte array
        byte[] m = ByteBuffer.allocate(4).putInt(i).array();
        // Declare a Datagram (UDP style) socket
        DatagramSocket aSocket = null;
        try {
            // Create a Datagram (UDP style) socket
            aSocket = new DatagramSocket();
            /*
            Build the packet holding the byte message from the console, length of the
message,
            destination address, and the destination port number.
            */
            DatagramPacket request = new DatagramPacket(m,  m.length, aHost, serverPort);
            // Send the Datagram request on the socket
            aSocket.send(request);
            // Prepare buffer for the reply
            byte[] replyBuffer = new byte[4];
            // Create a Datagram for the reply
            DatagramPacket reply = new DatagramPacket(replyBuffer, replyBuffer.length);
            // Wait and receive the reply
            aSocket.receive(reply);
            // Convert reply into integer
            ByteBuffer buffer = ByteBuffer.allocate(Integer.BYTES);
            buffer.put(reply.getData());
            buffer.rewind();
            sum = buffer.getInt();
            // Handle socket exceptions
        } catch (SocketException e) {
            throw new RuntimeException(e);
            // Handle general IO exceptions
        } catch (IOException e) {
            throw new RuntimeException(e);
            // Close the socket if not null
        } finally {if(aSocket != null) aSocket.close();}

        return sum;
```

```
    }
}
```

Project2Task2Server

```
/**
 * This program implements a UDP server.
 * @author Candice Chiang
 * Andrew id: wantienc
 * Last Modified: Feb 20, 2023
 */
// Import the necessary packages for UDP
import java.net.*;
import java.io.*;
import java.nio.ByteBuffer;

public class AddingServerUDP{
    // sum to record the total added values
    static int sum = 0;
    /**
     * Implement a UDP server.
     * @param args Array of strings from the console
     */
    public static void main(String args[]){
        // Announce the server starts running
        System.out.println("Server started");
        // Get the port number this server to listen on from user
        int serverPort = 6789;
        // Declare a Datagram (UDP style) socket
        DatagramSocket aSocket = null;
        // Prepare buffer for integer
        byte[] requestBuffer = new byte[4];
        try{
            // Create a new DatagramSocket and bind it to port number from user input
            aSocket = new DatagramSocket(serverPort);
            // Create a new DatagramPacket for receiving requests
            DatagramPacket request = new DatagramPacket(requestBuffer,
requestBuffer.length);
            // An infinite loop to wait for incoming datagrams
            while(true){
                // Receive a datagram
                aSocket.receive(request);
                // Convert the request byte array into integer
                ByteBuffer buffer = ByteBuffer.allocate(Integer.BYTES);
                buffer.put(request.getData());
                buffer.rewind();
                int numAdded = buffer.getInt();
                // Call add method to calculate
                add(numAdded);
                // Convert sum to byte array
                byte[] replySum = ByteBuffer.allocate(4).putInt(sum).array();
                /*
                Create a new DatagramPacket for sending replies
```

```
                with byte array of sum, array length, request address, and request port
number.
                */
                DatagramPacket reply = new DatagramPacket(replySum,
                            replySum.length, request.getAddress(), request.getPort());
                // Send a reply datagram back to the client
                aSocket.send(reply);
                // Print reply action
                System.out.println("Returning sum of " + sum + " to client");
                System.out.println();

            }
            // Handle socket exceptions
        }catch (SocketException e){System.out.println("Socket: " + e.getMessage());
            // Handle general IO exceptions
        }catch (IOException e) {System.out.println("IO: " + e.getMessage());
            // Close the socket if not null
        }finally {if(aSocket != null) aSocket.close();}
    }

    /**
     * Add integer i to sum
     * @param i integer requested from client
     * @return current sum
     */
    public static int add (int i) {
        // Print current adding action
        System.out.println("Adding " + i + " to " + sum);
        // Add i to sum
        sum += i;
        return sum;
    }
}
```

Project2Task2ClientConsole

```
Run:  AddingServerUDP ×   AddingClientUDP ×
/Library/Java/JavaVirtualMachines/
The UDP client is running.
Please enter server port:
6789

1
The server returned 1.
2
The server returned 3.
-3
The server returned 0.
4
The server returned 4.
5
The server returned 9.
halt!
Client side quitting.

Process finished with exit code 0
```

```
Run:  AddingServerUDP ×   AddingClientUDP ×
/Library/Java/JavaVirtualMachines/
The UDP client is running.
Please enter server port:
6789

6
The server returned 15.
7
The server returned 22.
-8
The server returned 14.
9
The server returned 23.
10
The server returned 33.
halt!
Client side quitting.

Process finished with exit code 0
```

Project2Task2ServerConsole

```
Run:  AddingServerUDP ×   AddingClientUDP ×
/Library/Java/JavaVirtualMachines/
Server started
Adding 1 to 0
Returning sum of 1 to client

Adding 2 to 1
Returning sum of 3 to client

Adding -3 to 3
Returning sum of 0 to client

Adding 4 to 0
Returning sum of 4 to client

Adding 5 to 4
Returning sum of 9 to client

Adding 6 to 9
Returning sum of 15 to client

Adding 7 to 15
Returning sum of 22 to client

Adding -8 to 22
Returning sum of 14 to client

Adding 9 to 14
Returning sum of 23 to client

Adding 10 to 23
Returning sum of 33 to client
```

Project 2Task 3

Project2Task3Client

```
/**
 * This program implements a UDP client.
 * @author Candice Chiang
 * Andrew id: wantienc
 * Last Modified: Feb 21, 2023
 */

// Import the necessary packages for UDP
import java.net.*;
import java.io.*;
import java.nio.ByteBuffer;
import java.util.Scanner;

public class RemoteVariableClientUDP{
    // Declare a Datagram (UDP style) socket
    static DatagramSocket aSocket = null;
    // Destination server port number
    static int serverPort;
    // Host name
    static InetAddress aHost;
    /**
     * Implement a UDP client.
     * @param args Array of strings giving message contents and server hostname
     */
    public static void main(String args[]){
        // Announce the client starts running
        System.out.println("The UDP client is running.");
        // Get the server side port number from user
        // For this project, use 6789
        Scanner readInput = new Scanner(System.in);
        System.out.println("Please enter server port: ");
        serverPort = readInput.nextInt();
        System.out.println();

        try {
            // Create a Datagram (UDP style) socket
            aSocket = new DatagramSocket();
            // Initialize choice
            int choice = 0;
            // Collect the IP address
            aHost = InetAddress.getByName("localhost");
            while (true){
                // Display menu
                System.out.println("1. Add a value to your sum.");
                System.out.println("2. Subtract a value from your sum.");
                System.out.println("3. Get your sum.");
                System.out.println("4. Exit client");
                if (readInput.hasNextInt()) {
                    // Get choice
```

```
                    choice = readInput.nextInt();
                    // num variable to store int for add or subtract choice
                    int num;
                    // id variable to store client's ID
                    int id;
                    // Initialize request string
                    String requestStr = "";

                    switch (choice) {
                        case 1: // If user selects to add
                            System.out.println("Enter value to add: ");
                            // Get num to add
                            num = readInput.nextInt();
                            System.out.println("Enter your ID:");
                            // Get id
                            id = readInput.nextInt();
                            // Concatenate request string
                            requestStr = id + "," + choice + "," + num;
                            break;
                        case 2: // If user selects to subtract
                            System.out.println("Enter value to subtract: ");
                            // Get num to subtract
                            num = readInput.nextInt();
                            System.out.println("Enter your ID:");
                            // Get id
                            id = readInput.nextInt();
                            // Concatenate request string
                            requestStr = id + "," + choice + "," + num;
                            break;
                        case 3: // If user selects to get
                            System.out.println("Enter your ID:");
                            // Get id
                            id = readInput.nextInt();
                            // Concatenate request string
                            requestStr = id + "," + choice;
                            break;
                        case 4: // If user selects to quit
                            System.out.println("Client side quitting. The remote variable
server is still running.");
                            break;
                    }
                    if (choice == 4) break; // Break the loop if clients requests to quit
                    // Call parseRequest method to communicate with server and get the
result
                    int result = parseRequest(requestStr);
                    // Print result to console
                    System.out.println("The result is " + result + ".");
                    System.out.println();
                }

            }
            // Handle unknown host exceptions
        } catch (UnknownHostException e) {
            throw new RuntimeException(e);
```

```
        } catch (SocketException e) { // Handle socket exceptions
            throw new RuntimeException(e);
        } finally {if(aSocket != null) aSocket.close();} // Close the socket if not null

    }

    public static int parseRequest (String requestStr) {
        // int result to record the reply sum
        int result;
        // Convert requestStr into byte array
        byte[] m = requestStr.getBytes();
        try {
            /*
            Build the packet holding the byte message from the console, length of the
message,
            destination address, and the destination port number.
            */
            DatagramPacket request = new DatagramPacket(m, m.length, aHost, serverPort);
            // Send the Datagram request on the socket
            aSocket.send(request);
            // Prepare buffer for the reply
            byte[] replyBuffer = new byte[4];
            // Create a Datagram for the reply
            DatagramPacket reply = new DatagramPacket(replyBuffer, replyBuffer.length);
            // Wait and receive the reply
            aSocket.receive(reply);
            // Convert reply into integer
            ByteBuffer buffer = ByteBuffer.allocate(Integer.BYTES);
            buffer.put(reply.getData());
            buffer.rewind();
            result = buffer.getInt();
        } catch (IOException e) { // Handle general IO exceptions
            throw new RuntimeException(e);
        }

        return result;
    }
}
```

Project2Task3Server

```
/**
 * This program implements a UDP server.
 * @author Candice Chiang
 * Andrew id: wantienc
 * Last Modified: Feb 22, 2023
 */
// Import the necessary packages for UDP
import java.net.*;
import java.io.*;
import java.nio.ByteBuffer;
import java.util.TreeMap;

public class RemoteVariableServerUDP{
    // TreeMap to store the sum for each id
    static TreeMap<Integer, Integer> idSumMap = null;
    /**
     * Implement a UDP server.
     * @param args Array of strings from the console
     */
    public static void main(String args[]){
        // Announce the server starts running
        System.out.println("Server started");
        // Get the port number this server to listen on from user
        int serverPort = 6789;
        // Declare a Datagram (UDP style) socket
        DatagramSocket aSocket = null;
        // Prepare buffer for integer
        byte[] requestBuffer = new byte[1000];
        try{
            // Create a new DatagramSocket and bind it to port number from user input
            aSocket = new DatagramSocket(serverPort);
            // Initialize the TreeMap
            idSumMap = new TreeMap<>();
            // Create a new DatagramPacket for receiving requests
            DatagramPacket request = new DatagramPacket(requestBuffer,
requestBuffer.length);
            // An infinite loop to wait for incoming datagrams
            while(true){
                // Receive a datagram
                aSocket.receive(request);
                // Convert the request byte array into string
                String requestStr = new String(request.getData()).substring(0,
request.getLength());
                // Split the requestStr to array [id, choice, (num)]
                String[] requestArr = requestStr.split(",");
                // Get id
                Integer id = Integer.parseInt(requestArr[0]);
                // Add id to map if the id hasn't requested before
                if (!idSumMap.containsKey(id)) {
                    idSumMap.put(id, 0);
                }
```

```
                // Get choice
                int choice = Integer.parseInt(requestArr[1]);
                int num;
                if (choice == 1) { // if the choice is adding
                    // Get the num to be added
                    num = Integer.parseInt(requestArr[2]);
                    // Call add method
                    add(id, num);
                } else if (choice == 2) { // if the choice is subtracting
                    // Get the num to be subtracted
                    num = Integer.parseInt(requestArr[2]);
                    // Call subtract method
                    subtract(id, num);
                } else { // Print getting action
                    System.out.println("ID: " + id + " - getting sum");
                }

                // Convert sum to byte array
                byte[] replySum =
ByteBuffer.allocate(4).putInt(idSumMap.get(id)).array();
                /*
                Create a new DatagramPacket for sending replies
                with byte array of sum, array length, request address, and request port
number.
                */
                DatagramPacket reply = new DatagramPacket(replySum,
                            replySum.length, request.getAddress(), request.getPort());
                // Send a reply datagram back to the client
                aSocket.send(reply);
                // Print reply action
                System.out.println("Returning sum of " + idSumMap.get(id) + " to
client");
                System.out.println();

            }
            // Handle socket exceptions
        }catch (SocketException e){System.out.println("Socket: " + e.getMessage());
            // Handle general IO exceptions
        }catch (IOException e) {System.out.println("IO: " + e.getMessage());
            // Close the socket if not null
        }finally {if(aSocket != null) aSocket.close();}
    }

    /**
     * Add num to sum for the id
     * @param id client ID
     * @param num int to be added
     * @return current sum for the id
     */
    public static int add (int id, int num) {
        // Print current adding action
        System.out.println("ID: " + id + " - adding " + num + " to " + idSumMap.get(id));
        // Add num to sum
        idSumMap.put(id, idSumMap.get(id) + num);
```

```
        return idSumMap.get(id);
    }

    /**
     * Subtract num to sum for the id
     * @param id client ID
     * @param num int to be subtracted
     * @return current sum for the id
     */
    public static int subtract (int id, int num) {
        // Print current subtracting action
        System.out.println("ID: " + id + " - subtracting " + num + " to " +
idSumMap.get(id));
        // Subtract num to sum
        idSumMap.put(id, idSumMap.get(id) - num);
        return idSumMap.get(id);
    }
}
```

Project2Task3ClientConsole

```
Run:  RemoteVariableServerUDP   RemoteVariableClientUDP
/Library/Java/JavaVirtualMachines/temurin-17.
The UDP client is running.
Please enter server port:
6789

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
1
Enter value to add:
3
Enter your ID:
1
The result is 3.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
2
Enter value to subtract:
1
Enter your ID:
1
The result is 2.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
1
The result is 2.
```

Run: RemoteVariableServerUDP × RemoteVariableClientUDP ×

```
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
1
Enter value to add:
9
Enter your ID:
2
The result is 9.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
2
Enter value to subtract:
2
Enter your ID:
2
The result is 7.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
2
The result is 7.
```

```
Run:  RemoteVariableServerUDP   RemoteVariableClientUDP
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
1
Enter value to add:
7
Enter your ID:
3
The result is 7.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
2
Enter value to subtract:
-2
Enter your ID:
3
The result is 9.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
3
The result is 9.
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

```
Run:  RemoteVariableServerUDP ×   RemoteVariableClientUDP ×
/Library/Java/JavaVirtualMachines/temurin-17.
The UDP client is running.
Please enter server port:
6789

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
1
The result is 2.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
2
The result is 7.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
3
The result is 9.
```

Project2Task3ServerConsole

```
Run: RemoteVariableServerUDP   RemoteVariableClientUDP
Server started
ID: 1 - adding 3 to 0
Returning sum of 3 to client

ID: 1 - subtracting 1 to 3
Returning sum of 2 to client

ID: 1 - getting sum
Returning sum of 2 to client

ID: 2 - adding 9 to 0
Returning sum of 9 to client

ID: 2 - subtracting 2 to 9
Returning sum of 7 to client

ID: 2 - getting sum
Returning sum of 7 to client

ID: 3 - adding 7 to 0
Returning sum of 7 to client

ID: 3 - subtracting -2 to 7
Returning sum of 9 to client

ID: 3 - getting sum
Returning sum of 9 to client

ID: 1 - getting sum
Returning sum of 2 to client

ID: 2 - getting sum
Returning sum of 7 to client

ID: 3 - getting sum
Returning sum of 9 to client
```

Project 2 Task 4

Project2Task4Client

```
/**
 * This program implements a TCP client.
 * @author Candice Chiang
 * Andrew id: wantienc
 * Last Modified: Feb 22, 2023
 */

// Import the necessary packages for TCP
import java.net.*;
import java.io.*;
import java.util.Scanner;

public class RemoteVariableClientTCP {
    // Declare a client socket
    static Socket clientSocket = null;
    // Declare a BufferedReader to read from client socket
    static BufferedReader in = null;
    // Declare a PrintWriter to write to client socket
    static PrintWriter out = null;
    // Destination server port number
    static int serverPort;
    // Host name
    static InetAddress aHost;
    /**
     * Implement a TCP client.
     * @param args Array of strings giving message contents and server hostname
     */
    public static void main(String args[]) {
        // Announce the client starts running
        System.out.println("The TCP client is running.");
        // Get the server side port number from user
        // For this project, use 6789
        Scanner readInput = new Scanner(System.in);
        System.out.println("Please enter server port: ");
        serverPort = readInput.nextInt();
        System.out.println();
        try {
            // Collect the IP address
            aHost = InetAddress.getByName("localhost");
            // Initialize socket
            clientSocket = new Socket(aHost, serverPort);
            // Initialize choice
            int choice = 0;
            while (true){
                // Display menu
                System.out.println("1. Add a value to your sum.");
                System.out.println("2. Subtract a value from your sum.");
                System.out.println("3. Get your sum.");
                System.out.println("4. Exit client");
```

```
                if (readInput.hasNextInt()) {
                    // Get choice
                    choice = readInput.nextInt();
                    // num variable to store int for add or subtract choice
                    int num;
                    // id variable to store client's ID
                    int id;
                    // Initialize request string
                    String requestStr = "";

                    switch (choice) {
                        case 1: // If user selects to add
                            System.out.println("Enter value to add: ");
                            // Get num to add
                            num = readInput.nextInt();
                            System.out.println("Enter your ID:");
                            // Get id
                            id = readInput.nextInt();
                            // Concatenate request string
                            requestStr = id + "," + choice + "," + num;
                            break;
                        case 2: // If user selects to subtract
                            System.out.println("Enter value to subtract: ");
                            // Get num to subtract
                            num = readInput.nextInt();
                            System.out.println("Enter your ID:");
                            // Get id
                            id = readInput.nextInt();
                            // Concatenate request string
                            requestStr = id + "," + choice + "," + num;
                            break;
                        case 3: // If user selects to get
                            System.out.println("Enter your ID:");
                            // Get id
                            id = readInput.nextInt();
                            // Concatenate request string
                            requestStr = id + "," + choice;
                            break;
                        case 4: // If user selects to quit
                            System.out.println("Client side quitting. The remote variable
server is still running.");
                            break;
                    }
                    if (choice == 4) break; // Break the loop if clients requests to quit
                    // Call parseRequest method to communicate with server and get the
result
                    int result = parseRequest(requestStr);
                    // Print result to console
                    System.out.println("The result is " + result + ".");
                    System.out.println();
                }

            }
```

```
        } catch (IOException e) {
            System.out.println("IO Exception:" + e.getMessage());
        } finally {
            try {
                // Close socket if not null
                if (clientSocket != null) {
                    clientSocket.close();
                }
            } catch (IOException e) {
                // ignore exception on close
            }
        }
    }

    public static int parseRequest (String requestStr) {
        // int result to record the reply sum
        String result;
        // Convert requestStr into byte array
        byte[] m = requestStr.getBytes();
        try {
            in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));;
            out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
            // Write request to server
            out.println(requestStr);
            out.flush();
            result = in.readLine(); // read a line of data from the stream
            // Handle IO exceptions
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        return Integer.parseInt(result);
    }
}
```

Project2Task4Server

```
/**
 * This program implements a TCP server.
 * @author Candice Chiang
 * Andrew id: wantienc
 * Last Modified: Feb 22, 2023
 */

// Import the necessary packages for TCP
import java.net.*;
import java.io.*;
import java.util.Scanner;
import java.util.TreeMap;

public class RemoteVariableServerTCP {
    // TreeMap to store the sum for each id
    static TreeMap<Integer, Integer> idSumMap = null;
    /**
     * Implement a TCP server.
     * @param args Array of strings from the console
     */
    public static void main(String args[]) {
        // Announce the server starts running
        System.out.println("Server started");
        // Port number this server to listen on
        int serverPort = 6789;
        // Initialize the TreeMap storing <id, sum>
        idSumMap = new TreeMap<>();
        // Declare client socket
        Socket clientSocket = null;
        try {
            // Create a new server socket
            ServerSocket listenSocket = new ServerSocket(serverPort);

            /*
             * Block waiting for a new connection request from a client.
             * When the request is received, "accept" it, and the rest
             * the tcp protocol handshake will then take place, making
             * the socket ready for reading and writing.
             */
            clientSocket = listenSocket.accept();
            // If we get here, then we are now connected to a client.

            // Set up "in" to read from the client socket
            Scanner in;
            in = new Scanner(clientSocket.getInputStream());

            // Set up "out" to write to the client socket
            PrintWriter out;
            out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
            // An infinite loop to wait for incoming requests
```

```
            while(true){
                if (in.hasNextLine()) { // if there exists a request
                    // Get request
                    String requestStr = in.nextLine();
                    // Split the requestStr to array [id, choice, (num)]
                    String[] requestArr = requestStr.split(",");
                    // Get id
                    Integer id = Integer.parseInt(requestArr[0]);
                    // Add id to map if the id hasn't requested before
                    if (!idSumMap.containsKey(id)) {
                        idSumMap.put(id, 0);
                    }
                    // Get choice
                    int choice = Integer.parseInt(requestArr[1]);
                    int num;
                    if (choice == 1) { // if the choice is adding
                        // Get the num to be added
                        num = Integer.parseInt(requestArr[2]);
                        // Call add method
                        add(id, num);
                    } else if (choice == 2) { // if the choice is subtracting
                        // Get the num to be subtracted
                        num = Integer.parseInt(requestArr[2]);
                        // Call subtract method
                        subtract(id, num);
                    } else { // Print getting action
                        System.out.println("ID: " + id + " - getting sum");
                    }
                    // Write sum result to socket
                    out.println(idSumMap.get(id));
                    out.flush();
                    // Print reply action
                    System.out.println("Returning sum of " + idSumMap.get(id) + " to
client");
                    System.out.println();

                } else { // Ready to accept another new connection request from client
                    clientSocket = listenSocket.accept();
                    in = new Scanner(clientSocket.getInputStream());
                    out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
                }

            }

            // Handle exceptions
        } catch (IOException e) {
            System.out.println("IO Exception:" + e.getMessage());

            // If quitting (typically by you sending quit signal) clean up sockets
        } finally {
            try {
                if (clientSocket != null) { // Close socket if not null
                    clientSocket.close();
```

```
                }
            } catch (IOException e) {
                // ignore exception on close
            }
        }
    }

    /**
     * Add num to sum for the id
     * @param id client ID
     * @param num int to be added
     * @return current sum for the id
     */
    public static int add (int id, int num) {
        // Print current adding action
        System.out.println("ID: " + id + " - adding " + num + " to " + idSumMap.get(id));
        // Add num to sum
        idSumMap.put(id, idSumMap.get(id) + num);
        return idSumMap.get(id);
    }

    /**
     * Subtract num to sum for the id
     * @param id client ID
     * @param num int to be subtracted
     * @return current sum for the id
     */
    public static int subtract (int id, int num) {
        // Print current subtracting action
        System.out.println("ID: " + id + " - subtracting " + num + " to " +
idSumMap.get(id));
        // Subtract num to sum
        idSumMap.put(id, idSumMap.get(id) - num);
        return idSumMap.get(id);
    }
}
```

Project2Task4ClientConsole

```
Run:  RemoteVariableClientTCP   RemoteVariableServerTCP
/Library/Java/JavaVirtualMachines/temurin-17
The TCP client is running.
Please enter server port:
6789

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
1
Enter value to add:
2
Enter your ID:
1
The result is 2.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
2
Enter value to subtract:
1
Enter your ID:
1
The result is 1.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
1
The result is 1.
```

```
Run:  RemoteVariableClientTCP  RemoteVariableServerTCP
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
1
Enter value to add:
6
Enter your ID:
2
The result is 6.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
2
Enter value to subtract:
2
Enter your ID:
2
The result is 4.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
2
The result is 4.
```

```
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
1
Enter value to add:
-1
Enter your ID:
3
The result is -1.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
2
Enter value to subtract:
1
Enter your ID:
3
The result is -2.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

```
Run:  RemoteVariableClientTCP   RemoteVariableServerTCP
/Library/Java/JavaVirtualMachines/temurin-17.
The TCP client is running.
Please enter server port:
6789

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
1
The result is 1.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
2
The result is 4.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Enter your ID:
3
The result is -2.
```

Project2Task4ServerConsole

```
Run:  RemoteVariableClientTCP   RemoteVariableServerTCP
/Library/Java/JavaVirtualMachines/temurin-17.
Server started
ID: 1 - adding 2 to 0
Returning sum of 2 to client

ID: 1 - subtracting 1 to 2
Returning sum of 1 to client

ID: 1 - getting sum
Returning sum of 1 to client

ID: 2 - adding 6 to 0
Returning sum of 6 to client

ID: 2 - subtracting 2 to 6
Returning sum of 4 to client

ID: 2 - getting sum
Returning sum of 4 to client

ID: 3 - adding -1 to 0
Returning sum of -1 to client

ID: 3 - subtracting 1 to -1
Returning sum of -2 to client

ID: 1 - getting sum
Returning sum of 1 to client

ID: 2 - getting sum
Returning sum of 4 to client

ID: 3 - getting sum
Returning sum of -2 to client
```

Project 2 Task 5

Project2Task5Client

```
/**
 * This program implements a TCP client.
 * @author Candice Chiang
 * Andrew id: wantienc
 * Last Modified: Feb 24, 2023
 */

// Import the necessary packages
import java.io.*;
import java.math.BigInteger;
import java.net.InetAddress;
import java.net.Socket;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Random;
import java.util.Scanner;

/** SigningClientTCP.java provides capabilities to sign add, subtract, or get requests.
 *
 *  For signing: the SigningClientTCP object is constructed with RSA
 *  keys (e,d,n). These keys are randomly created with 2048 bits.
 *  Then, a caller can sign a message - the string returned by the sign
 *  method is evidence that the signer has the associated private key.
 */


public class SigningClientTCP {
    // RSA keys
    private BigInteger e,d,n;
    // Declare a client socket
    static Socket clientSocket = null;
    // Length to take for calculating ID
    static private final int hash_length_id = 20;
    // Declare a BigInteger id
    static private BigInteger id;
    // Declare a Scanner to read input from console
    static Scanner readInput = null;
    // Declare a BufferedReader to read from client socket
    static BufferedReader in = null;
    // Declare a PrintWriter to write to client socket
    static PrintWriter out = null;
    // Destination server port number
    static int serverPort;
    // Host name
    static InetAddress aHost;

    /** A ShortMessageSign object may be constructed with RSA's e, d, and n.
     *  The holder of the private key (the signer) would call this
```

```
     *  constructor. Only d and n are used for signing.
     */
    public SigningClientTCP (BigInteger e, BigInteger d, BigInteger n) {
        this.e = e;
        this.d = d;
        this.n = n;
    }




    public static void main(String args[]) throws Exception {
        // Announce the client starts running
        System.out.println("The TCP client is running.");
        // Get the server side port number from user
        // For this project, use 6789
        readInput = new Scanner(System.in);
        System.out.println("Please enter server port: ");
        // Get server port
        serverPort = readInput.nextInt();
        System.out.println();
        // Collect the IP address
        aHost = InetAddress.getByName("localhost");
        // Initialize socket
        clientSocket = new Socket(aHost, serverPort);
        // Code refer to RSAExample.java
        // Each public and private key consists of an exponent and a modulus
        BigInteger n; // n is the modulus for both the private and public keys
        BigInteger e; // e is the exponent of the public key
        BigInteger d; // d is the exponent of the private key

        Random rnd = new Random();

        // Step 1: Generate two large random primes.
        // We use 400 bits here, but best practice for security is 2048 bits.
        // Change 400 to 2048, recompile, and run the program again and you will
        // notice it takes much longer to do the math with that many bits.
        BigInteger p = new BigInteger(2048,100,rnd);
        BigInteger q = new BigInteger(2048,100,rnd);

        // Step 2: Compute n by the equation n = p * q.
        n = p.multiply(q);

        // Step 3: Compute phi(n) = (p-1) * (q-1)
        BigInteger phi =
(p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));

        // Step 4: Select a small odd integer e that is relatively prime to phi(n).
        // By convention the prime 65537 is used as the public exponent.
        e = new BigInteger ("65537");

        // Step 5: Compute d as the multiplicative inverse of e modulo phi(n).
        d = e.modInverse(phi);
        String publicKey = "(" + e + "," + n + ")";
```

```
        String privateKey = "(" + d + "," + n + ")";
        System.out.println("Public Key (e,n): " + publicKey);  // Step 6: (e,n) is the
RSA public key
        System.out.println("Private Key (d,n): " + privateKey);  // Step 7: (d,n) is the
RSA private key

        SigningClientTCP sov = new SigningClientTCP(e,d,n);
        // Generate id for the current session
        sov.generateID(String.valueOf(e) + String.valueOf(n));
        // Declare a request str to send
        String requestStr = "";
        // Declare a reply str for the reply
        String reply;

        while (true) {
            // Get the choice from user
            int choice = sov.getChoice();
            if (choice != 4) { // Choices other than quitting
                // Get the requestStr including id, choice, operand, public key,
signature
                requestStr = sov.getRequestStr(choice, e, n);
                // Print the message before signing
                System.out.println("Clear Message: " + requestStr);
                // Sign the request
                String signedVal = sov.sign(requestStr);
                // Print the signature
                System.out.println("Signed Message: " + signedVal);
                // Concatenate the message and the signature
                requestStr = requestStr + ";" + signedVal;
                // Send request to server
                reply = sov.send(requestStr);
                // Print result to console
                System.out.println("The result is " + reply + ".");
                System.out.println();
            } else { // Client requests quiiting
                System.out.println("Client side quitting. The remote variable server is
still running.");
                break;
            }
        }

    }

    /**
     * Hash function using SHA-256
     * @param hashStr
     * @return the hashed string in byte array
     * @throws NoSuchAlgorithmException
     * @throws UnsupportedEncodingException
     */
    private byte[] h(String hashStr) throws NoSuchAlgorithmException,
UnsupportedEncodingException {
        // compute the digest with SHA-256
        byte[] bytesOfMessage = hashStr.getBytes("UTF-8");
```

```
    MessageDigest md = MessageDigest.getInstance("SHA-256");
    byte[] bigDigest = md.digest(bytesOfMessage);
    return bigDigest;
}

/**
 * Generate a unique id with the last 20 byte of the public key
 * @param publicKey e+n
 */
private void generateID(String publicKey){
    try {
        // Get the hashed value
        byte[] hash_value = h(publicKey);
        // Get the last 20 bytes
        byte[] id_byte = new byte[hash_length_id];
        for(int i = 0; i < hash_length_id; i++){
            id_byte[hash_length_id-i-1] = hash_value[hash_value.length - i - 1];
        }
        id = new BigInteger(id_byte);
        System.out.println("ID: " + id);
    } catch(NoSuchAlgorithmException e) {
        System.out.println("No Hash available" + e);
    } catch (UnsupportedEncodingException ex) {
        throw new RuntimeException(ex);
    }

}
// Code refer to ShortMessageSign.java
/**
 * Signing proceeds as follows:
 * 1) Get the bytes from the string to be signed.
 * 2) Compute a SHA-1 digest of these bytes.
 * 3) Copy these bytes into a byte array that is one byte longer than needed.
 *    The resulting byte array has its extra byte set to zero. This is because
 *    RSA works only on positive numbers. The most significant byte (in the
 *    new byte array) is the 0'th byte. It must be set to zero.
 * 4) Create a BigInteger from the byte array.
 * 5) Encrypt the BigInteger with RSA d and n.
 * 6) Return to the caller a String representation of this BigInteger.
 * @param message a sting to be signed
 * @return a string representing a big integer - the encrypted hash.
 * @throws Exception
 */
public String sign(String message) throws Exception {
    // Get the hashed message
    byte[] bigDigest = h(message);

    // Get the signed value
    // we add a 0 byte as the most significant byte to keep
    // the value to be signed non-negative.
    byte[] messageDigest = new byte[bigDigest.length + 1];
    messageDigest[0] = 0;   // most significant set to 0
    for (int i = 1; i < bigDigest.length; i++) {
        messageDigest[i] = bigDigest[i-1]; // take a byte from SHA-256
```

```
    }

    // From the digest, create a BigInteger
    BigInteger m = new BigInteger(messageDigest);

    // encrypt the digest with the private key
    BigInteger c = m.modPow(d, n);

    // return this as a big integer string
    return c.toString();
}

/**
 * Get user choice.
 * 1: Add, 2: Subtract, 3: Get
 * @return choice number
 */
private int getChoice() {
    readInput = new Scanner(System.in);
    // Initialize choice
    int choice = 0;
    // Display menu
    System.out.println("1. Add a value to your sum.");
    System.out.println("2. Subtract a value from your sum.");
    System.out.println("3. Get your sum.");
    System.out.println("4. Exit client");
    if (readInput.hasNextInt()) {
        // Get choice
        choice = readInput.nextInt();
    }
    return choice;
}

/**
 * Return request string in format "id,choice,operand,e,n"
 * @param choice choice number
 * @param e e
 * @param n n
 * @return full request string without signature
 */
private String getRequestStr(int choice, BigInteger e, BigInteger n) {
    readInput = new Scanner(System.in);
    // Initialize request string
    String requestStr = "";

    // num variable to store int for add or subtract choice
    int num;

    switch (choice) {
        case 1: // If user selects to add
            System.out.println("Enter value to add: ");
            // Get num to add
            num = readInput.nextInt();
            // Concatenate request string
```

```
                requestStr = id + "," + choice + "," + num + "," + e + "," + n;
                break;
            case 2: // If user selects to subtract
                System.out.println("Enter value to subtract: ");
                // Get num to subtract
                num = readInput.nextInt();
                // Concatenate request string
                requestStr = id + "," + choice + "," + num + "," + e + "," + n;
                break;
            case 3: // If user selects to get
                // Concatenate request string
                requestStr = id + "," + choice + "," + 0 + "," + e + "," + n;
                break;
        }
        return requestStr;
    }

    /**
     * Communicate with server
     * @param requestStr request string including signature
     * @return reply from server
     */
    private String send(String requestStr) {
        // int result to record the reply sum
        String result;
        // Convert requestStr into byte array
        byte[] m = requestStr.getBytes();
        try {
            in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));;
            out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
            // Write request to server
            out.println(requestStr);
            out.flush();
            result = in.readLine(); // read a line of data from the stream
            // Handle IO exceptions
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        return result;
    }
}
```

Project2Task5Server

```
/**
 * This program implements a TCP server.
 * @author Candice Chiang
 * Andrew id: wantienc
 * Last Modified: Feb 24, 2023
 */

// Import the necessary packages
import java.io.*;
import java.math.BigInteger;
import java.net.ServerSocket;
import java.net.Socket;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Scanner;
import java.util.TreeMap;

/** VerifyingServerTCP.java provides capabilities to verify messages.
 *  VerifyingServerTCP has two private members: RSA e and n.
 *
 *  For verification: the object is constructed with keys (e and n). The verify
 *  method is called with two parameters - the string to be checked and the
 *  evidence that this string was indeed manipulated by code with access to the
 *  private key d.
 */

public class VerifyingServerTCP {
    // RSA keys
    private BigInteger e,n;
    // TreeMap to store the sum for each id
    static TreeMap<BigInteger, Integer> idSumMap = null;
    //
    static private final int hash_length_id = 20;

    /** For verifying, a SignOrVerify object may be constructed
     *  with a RSA's e and n. Only e and n are used for signature verification.
     */
    public VerifyingServerTCP (BigInteger e, BigInteger n) {
        this.e = e;
        this.n = n;
    }
    // Code refer to ShortMessageVerify.java
    /**
     * Verifying proceeds as follows:
     * 1) Decrypt the encryptedHash to compute a decryptedHash
     * 2) Hash the messageToCheck using SHA-256 (be sure to handle
     *    the extra byte as described in the signing method.)
     * 3) If this new hash is equal to the decryptedHash, return true else false.
     *
     * @param messageToCheck  a normal string (4 hex digits) that needs to be verified.
     * @param encryptedHashStr integer string - possible evidence attesting to its
origin.
```

```
     * @return true or false depending on whether the verification was a success
     * @throws Exception
     */
    private boolean verify(String messageToCheck, String encryptedHashStr) throws
Exception  {
        // Take the encrypted string and make it a big integer
        BigInteger encryptedHash = new BigInteger(encryptedHashStr);
        // Decrypt it
        BigInteger decryptedHash = encryptedHash.modPow(e, n);

        // Get the bytes from messageToCheck
        byte[] bytesOfMessageToCheck = messageToCheck.getBytes("UTF-8");

        // compute the digest of the message with SHA-256
        MessageDigest md = MessageDigest.getInstance("SHA-256");

        byte[] messageToCheckDigest = md.digest(bytesOfMessageToCheck);

        // messageToCheckDigest is a full SHA-256 digest
        // take two bytes from SHA-256 and add a zero byte
        byte[] extraByte = new byte[messageToCheckDigest.length + 1];
        extraByte[0] = 0;
        for (int i = 1; i < messageToCheckDigest.length; i++) {
            extraByte[i] = messageToCheckDigest[i - 1];
        }

        // Make it a big int
        BigInteger bigIntegerToCheck = new BigInteger(extraByte);

        // inform the client on how the two compare
        if(bigIntegerToCheck.compareTo(decryptedHash) == 0) {
            System.out.println("Signature Verified: Pass" );
            System.out.println("Clear Message: " + messageToCheck);
            System.out.println("Signed Message: " + encryptedHashStr);
            return true;
        }
        else {
            System.out.println("Signature Verified: Fail" );
            return false;
        }
    }

    /**
     * Verify if the public key hash to the ID
     * @param id id passed by client
     * @param publicKey e+n
     * @return true if the public key hash to the ID, and false otherwise
     */
    private boolean verifyID(BigInteger id, String publicKey) {
        // compute the digest with SHA-256
        byte[] bytesOfMessage;
        try {
            bytesOfMessage = publicKey.getBytes("UTF-8");
            MessageDigest md = MessageDigest.getInstance("SHA-256");
```

```
            byte[] hash_value = md.digest(bytesOfMessage);
            byte[] id_byte = new byte[hash_length_id];
            for(int i = 0; i < hash_length_id; i++){
                id_byte[hash_length_id-i-1] = hash_value[hash_value.length - i - 1];
            }
            BigInteger calculatedID = new BigInteger(id_byte);
            if (calculatedID.compareTo(id) == 0) {
                System.out.println("ID Verified: Pass" );
                return true;
            } else {
                System.out.println("ID Verified: Fail" );
                return false;
            }
        } catch (UnsupportedEncodingException ex) {
            throw new RuntimeException(ex);
        } catch (NoSuchAlgorithmException ex) {
            throw new RuntimeException(ex);
        }
    }


    public static void main(String args[]) throws Exception {
        // Announce the server starts running
        System.out.println("Server started");
        // Port number this server to listen on
        int serverPort = 6789;
        // Initialize the TreeMap storing <id, sum>
        idSumMap = new TreeMap<>();
        // Declare client socket
        Socket clientSocket = null;
        try {
            // Create a new server socket
            ServerSocket listenSocket = new ServerSocket(serverPort);
            /*
             * Block waiting for a new connection request from a client.
             * When the request is received, "accept" it, and the rest
             * the tcp protocol handshake will then take place, making
             * the socket ready for reading and writing.
             */
            clientSocket = listenSocket.accept();
            // If we get here, then we are now connected to a client.

            // Set up "in" to read from the client socket
            Scanner in;
            in = new Scanner(clientSocket.getInputStream());

            // Set up "out" to write to the client socket
            PrintWriter out;
            out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));


            VerifyingServerTCP verifySig = null;
            // An infinite loop to wait for incoming requests
```

```
            while (true) {
                if (in.hasNextLine()) { // if there exists a request
                    // Get request
                    String requestStr = in.nextLine();
                    // Split the requestStr to array [clear message, signature]
                    String[] m = requestStr.split(";");
                    // Split the clear message to array [id, choice, operand, e, n]
                    String[] element = m[0].split(",");
                    // Get the elements
                    BigInteger id = new BigInteger(element[0]);
                    int choice = Integer.parseInt(element[1]);
                    int num = Integer.parseInt(element[2]);
                    BigInteger e = new BigInteger(element[3]);
                    BigInteger n = new BigInteger(element[4]);
                    verifySig = new VerifyingServerTCP(e, n);
                    String publicKey = String.valueOf(e) + String.valueOf(n);
                    // Verify the request
                    if (element.length == 5 && verifySig.verifyID(id, publicKey) &&
verifySig.verify(m[0], m[1])) {
                        // Add id to map if the id hasn't requested before
                        if (!idSumMap.containsKey(id)) {
                            idSumMap.put(id, 0);
                        }
                        System.out.println("ID: " + id);
                        if (choice == 1) { // if the choice is adding
                            // Call add method
                            add(id, num);
                        } else if (choice == 2) { // if the choice is subtracting
                            // Call subtract method
                            subtract(id, num);
                        } else { // Print getting action
                            System.out.println("Getting sum...");
                        }
                        // Write sum result to socket
                        out.println(idSumMap.get(id));
                        out.flush();
                        // Print reply action
                        System.out.println("Returning sum of " + idSumMap.get(id) + " to
client");
                        System.out.println();
                    } else { // Reply error if the request is not valid
                        out.println("Error in request");
                        out.flush();
                        // Print reply action
                        System.out.println("Error in request");
                        System.out.println();
                    }
                } else { // Ready to accept another new connection request from client
                    clientSocket = listenSocket.accept();
                    in = new Scanner(clientSocket.getInputStream());
                    out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
                }
```

```
            }
        } catch (IOException e) {
            System.out.println("IO Exception:" + e.getMessage());

            // If quitting (typically by you sending quit signal) clean up sockets
        } finally {
            try {
                if (clientSocket != null) { // Close socket if not null
                    clientSocket.close();
                }
            } catch (IOException e) {
                // ignore exception on close
            }
        }


    }

    /**
     * Add num to sum for the id
     * @param id client ID
     * @param num int to be added
     * @return current sum for the id
     */
    public static int add (BigInteger id, int num) {
        // Print current adding action
        System.out.println("Adding " + num + " to " + idSumMap.get(id));
        // Add num to sum
        idSumMap.put(id, idSumMap.get(id) + num);
        return idSumMap.get(id);
    }

    /**
     * Subtract num to sum for the id
     * @param id client ID
     * @param num int to be subtracted
     * @return current sum for the id
     */
    public static int subtract (BigInteger id, int num) {
        // Print current subtracting action
        System.out.println("Subtracting " + num + " to " + idSumMap.get(id));
        // Subtract num to sum
        idSumMap.put(id, idSumMap.get(id) - num);
        return idSumMap.get(id);
    }

}
```

Project2Task5ClientConsole

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -
javaagent:/Applications/IntelliJ
IDEA.app/Contents/lib/idea_rt.jar=54690:/Applications/IntelliJ IDEA.app/Contents/bin -
Dfile.encoding=UTF-8 -classpath
/Users/wantienchiang/IdeaProjects/DS/Project2Task5/out/production/Project2Task5
SigningClientTCP
The TCP client is running.
Please enter server port:
6789

Public Key (e,n):
(65537,446013130607378921483381433319880598399223812078244586243586647893874717313415713993988660944502473733777326512200494130449600843184007188463149836820734043101946765328133
425324305967455919056880354069778261039759583009974585207920276968301123919602289736369082209674331493725863942739958651283015559304522870257356991211412939748457124874000024645555027263043852690914424161082373845499317910748374967523586543285043142213534285268768000
194626583474234347024441878880340815592819282809448399797909206341543881057681127940242864964236459298590151064173393299087819025192106111402564122374564397912837849436900522247006838435375480456540374565652479277031901135533561160495319726362039352982633654055056245
839667558376405719514103007378428664632125071336438696607606729150720230465566511088783367637834705738993800235540021591136375534792245314288767847732566096004429668349514383306892922326806944826751716373593957012209218137570601492219025679827762150283860749832425855
880588661258032068369664810440369773948994351998385157630322455740187808782547674224611530746449617510532936716165151326559827378920691694196452863407579361857730267168845225720184524548317896768555451256829560152130748388604293473386797295774207098085217055489)
Private Key (d,n):
(103852180799679606052098824670970260029787072528709162550576502538421474681519199773383996307629396358964279759161687908062024640538748335992609770204379228492846905395537117512
991273072879820681358669222214374578196083620125582081319354506235424464515332972095068734154730827726074831191067924665722187268550591685119640602430086233055000644527153991939149397274700754598359676473211808895605810766467864106239994655818090216720222185935928864636217965670317423487064009656315461834622765500113158894804033746203521888587639136966206087689343206207780123926088776085637602259181883951164011973574467943030762805277761827514661408053349497362462131834299956642903361937017500259012211306946573865606536358883791100681103092485094773026941316738910550649020061537479396630222270789333108408411128868432978163452595766907384098690789208717206061210722323454639649348041398437134287385505481237448816231453419660250775929172319512231575389146795363364156170561285083557941727971666757387706981010159791596443232672735812972257794981931133756308824079284680324116880280527489170734095650686262284105073637845735748496785661158790787101044616886869870967830724973586008491650407034921466957537115766413229139009430910771669091473573969934460131306073789214833814333198805983992238120782445862435866478938747173134157139939886609445024737337773265122004941304496008431840071884631498368207340431019467653281334253243059674559190568803540697782610397595830099745852079202769683011239196022897363690822096743314937258639427399586512830155593045228702573569912114129397484571248740000246455550272630438526909
```

1442416108237384549931791074837496752358654328504314221353428526876800019462658347423434702444187888034081559281928280944839979790920634154388105768112794024286496423645929859015106417339329908781902519210611140256412237456439791283784943690052224700683843537548045654037456565245792770319011355335611604953197263620393529826336540550562458396675583764057195141030073784286640632125071336438696607606729150720230465566511088783367637834705738993800235540021591136375534792245314288767847732566096004429668349514383306892922326806944826751716373593957012209218137570601492219025679827762150283860749832425855880588661258032068369664810440369773948994351998385157630322455740187808782547674224611530746449617510532936716165151326559827378920691694196452863407579361857730267168845225720184524548317896768555451256829560152130748388604293473386797295774207098085217055489)
ID: 575515818954178421565834089266231852771230696305
1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
1
Enter value to add:
10
Clear Message:
575515818954178421565834089266231852771230696305,1,10,65537,4460131306073789214833814333198805983992238120782445862435866478938747173134157139939886609445024737337773265122004941304496008431840071884631498368207340431019467653281334253243059674559190568803540697782610397595830099745852079202769683011239196022897363690822096743314937258639427399586512830155593045228702573569912114129397484571248740000246455550272630438526909144241610823738454993179107483749675235865432850431422135342852687680001946265834742343470244418788803408155928192828094483997979092063415438810576811279402428649642364592985901510641733932990878192502519210611140256412237456439791283784943690052224700683843537548045654037456565247927703190113553356116049531972636203935298263365405505624583966755837640571951410300737842866403212507133643869660760672915072023046556651108878336763783470573899380023554002159113637553479224531428876784773256609600442966834951438330689292232680694482675171637359395702212092181375706014922190256798277621502838607498324258558805886612580320683696648104403697739489943519983851576303224557401878087825476742246115307464496175105329367161651513265598273789206916941964528634075793618577302671688452257201845245483178967685554512568295601521307483886042934733867972957742070980852170554891)
Signed Message:
4427518580150318476523108956161696219477126715706348479804866855619733297991691868999884650960602976130377204570227856573226109211142386908552587559543263668228787439771697397111354958269649302611844181879674604425796932999293459441473333500966961040370755968347485445984683630102457272520580110170271969897620257960583220967355687768343069890955657322356473973786205975401763766896412766838681463038448013066658798321068611380855746713198688357513416766304113762036224942557087545622990518798899270072071976626915392375866542664948954546123644525011177787996959101865923681718872064871540721569602202062409287470643878285461851865468795235914570842604771180137891862667944806860313079424331317085789066014830107963219732625859949120946614901231140822227331312458561787764854017957722838661349290660621432073387561968144315186073437789396184486243561876831826511223761020939673756133531279703286081793267463470634428686831717707705808543021525910942608912228014271298905678788621978351903612092360122301375806313832582510971223719259779352145991916183484259164816977945911802738244364713016876815741454970542996481829453411496707179306767336057347557069534088003545895873980611328421121414925152982194527732882179916720750016720635921 0
The result is 10.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.

```
4. Exit client
2
Enter value to subtract:
2
Clear Message:
575515818954178421565834089266231852771230696305,2,2,65537,446013130607378921483381433319
880598399223812078244586243586647893874717313415713993988660944502473733773265122004941 3
044960084318400718846314983682073404310194676532813342532430596745591905688035406977826 10
397595830099745852079202769683011239196022897363690822096743314937258639427399586512830 15
559304522870257356991211412939748457124874000024645555027263043852690914424161082373845 49
931791074837496752358654328504314221353428526876800019462658347423434702444187888034081 55
928192828094483997979092063415438810576811279402428649642364592985901510641733932990878 19
025192106111402564122374564397912837849436900522247006838435375480456540374565652479277 03
190113553356116049531972636203935298263365405505624583966755837640571951410300737842866 46
321250713364386966076067291507202304655665110887833676378347057389938002355400215911363 75
534792245314288767847732566096004429668349514383306892922326806944826751716373593957012 20
921813757060149221902567982776215028386074983242585588058866125803206836966481044036977 39
489943519983851576303224557401878087825476742246115307464496175105329367161651513265598 27
378920691694196452863407579361857730267168845225720184524548317896768555451256829560152 13
074838860429347338679729577420709808521705548 9
Signed Message:
396344054033908993475531176768991889301356708811249272249388276155438367188222984309631 52
091634536387842792194508930719898849352989052972639029939169835428054993474696210637211 38
987454442777779588646920169950338810807419949677070405929202195800387082292671576521619 05
180971557505217056586121234435008672436263873818009003765553606244670151482399154295744 34
234367086705931345891891253778083837956430492749149357259331139764701688543115341892629 71
019543676916282054549886203307179697926417376985120667121588637436437044389768392059263 17
351086916852697644734520128158039066712759855441555238580846500943162654347191441037413 82
525939741074964303027059666248401333238831042033501213508823380800451307527912981401600 12
560837015039138853676967306869144853347919741714477124005049836779169026926195260681545 43
630733542174172673841433494906900955821238044691669180166235032419925087317433244327844 57
347985141219315919410246439986152575955521276429965872208543538596457632572212331015604 91
957179968207091053206838599182923948437240434437293430147733436635826953942547685018301 42
997879200849767800726433960475381523323074303498959373921547253425649139515660521329699 25
598415801749290577753708589503018963737695623761512784238838777102877234578 0
The result is 8.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
3
Clear Message:
575515818954178421565834089266231852771230696305,3,0,65537,446013130607378921483381433319
880598399223812078244586243586647893874717313415713993988660944502473733773265122004941 3
044960084318400718846314983682073404310194676532813342532430596745591905688035406977826 10
397595830099745852079202769683011239196022897363690822096743314937258639427399586512830 15
559304522870257356991211412939748457124874000024645555027263043852690914424161082373845 49
931791074837496752358654328504314221353428526876800019462658347423434702444187888034081 55
928192828094483997979092063415438810576811279402428649642364592985901510641733932990878 19
025192106111402564122374564397912837849436900522247006838435375480456540374565652479277 03
190113553356116049531972636203935298263365405505624583966755837640571951410300737842866 46
321250713364386966076067291507202304655665110887833676378347057389938002355400215911363 75
534792245314288767847732566096004429668349514383306892922326806944826751716373593957012 20
```

```
92181375706014922190256798277621502838607498324258558805886612580320683696648104403697739
48994351998385157630322455740187808782547674224611530746449617510532936716165151326559827
37892069169419645286340757936185773026716884522572018452454831789676855545125682956015213
0748388604293473386797295774207098085217055489
Signed Message:
12014041557304991410762232727815481826804454732220871244503148876615841442426412704840551
42028986633982359704037024378038504144012184498813151839869304655551198129571344741618903
38281301627157189614740089451591881587814401015275708576821048600134472131840433312806908
07283454974284511141766789458412097908548974726451231374328455657717307905632928682860361
29803228806100488906391707411342392563044094738317529254014410066546842287944395461229159
51218652275162640366140127170722233732691583759429899731315779493842205472926791650779282
04541729343781748463433308213834038745615928454202212043795177523641723880939140973194277
65981509634858486425043948807326040277390256464306491632738076860145626789459763751694382
36398673223927173242052924664587228749267165434927018449729127534024874936861980440093893
30283035379654226574816506832565300288854237315205576319083149825893134468709538087182731
84794054261911613321714385228808711298072181020722951078850488961110196962660997213777748
13047705939040529932910936466306675391862414299349289740804486709999040370956489585844398
68399691345356280981828579775426564375910991356769227008493116153182440782142443461334947
605374034417538994167762465707414311951325642346516235283871135717563761994 5
The result is 8.

1. Add a value to your sum.
2. Subtract a value from your sum.
3. Get your sum.
4. Exit client
4
Client side quitting. The remote variable server is still running.

Process finished with exit code 0
```

Project2Task5ServerConsole

```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -
javaagent:/Applications/IntelliJ
IDEA.app/Contents/lib/idea_rt.jar=54687:/Applications/IntelliJ IDEA.app/Contents/bin -
Dfile.encoding=UTF-8 -classpath
/Users/wantienchiang/IdeaProjects/DS/Project2Task5/out/production/Project2Task5
VerifyingServerTCP
Server started
ID Verified: Pass
Signature Verified: Pass
Clear Message:
57551581895417842156583408926623185277123069630 5,1,10,65537,44601313060737892148338143331
98805983992238120782445862435866478938747173134157139939886609445024737337773265122004941
30449600843184007188463149836820734043101946765328133425324305967455919056880354069778261
03975958300997458520792027696830112391960228973636908220967433149372586394273995865128301
55593045228702573569912114129397484571248740000246455550272630438526909144241610823738454
99317910748374967523586543285043142213534285268768000194626583474234347024441878880340815
59281928280944839979790920634154388105768112794024286496423645929859015106417339329908781
90251921061114025641223745643979128378494369005222470068384353754804565403745656524792770
31901135533561160495319726362039352982633654055056245839667558376405719514103007378428664
63212507133643869660760672915072023046556651108878336763783470573899380023554002159113637
55347922453142887678477325660960044296683495143833068929223268069448267517163735939570122
```

```
0921813757060149221902567982776215028386074983242585588058866125803206836966481044036977 3
9489943519983851576303224557401878087825476742246115307464496175105329367161651513265598 2
7378920691694196452863407579361857730267168845225720184524548317896768555451256829560152 1
3074838860429347338679729577420709808521705548 9
Signed Message:
4427518580150318476523108956161696219477126715706348479804866855619733297991691868999884 6
5096060297613037720457022785657322610921114238690855258755954326366822878743977169739711 1
3549582696493026118441818796746044257969329992934594414733335009669610403707559683474854 4
5984683630102457272520580110170271969897620257960583220967355687768343069890955657322356 4
7397862059754017637668964127668386814630384480130666587983210686113808557467131986883575 1
3416766304113762036224942557087545622990518798899270072071976626915392375866542664948954 5
4612364452501117778799695910186592368171887206487154072156960220206240928747064387828546 1
8518654687952359145708426047711801378918626679448068603130794243313170857890660148301079 6
3219732625859949120946614901231140822227331312458561787764854017957722838661349290660621 4
3207338756196814431518607343778939618448624356187683182651122376102093967375613353127970 3
2860817932674634706344286868317177077058085430215259109426089122280142712989056787886219 7
8351903612092360122301375806313832582510971223719259779352145991916183484259164816977945 9
1180273824436471301687681574145497054299648182945341149670717930676733605734755706953408 8
0035458958739806113284211214149251529821945277328821799167207500167206359210
ID: 5755158189541784215658340892662318527712306963 05
Adding 10 to 0
Returning sum of 10 to client

ID Verified: Pass
Signature Verified: Pass
Clear Message:
5755158189541784215658340892662318527712306963 05,2,2,65537,4460131306073789214833814333 19
8805983992238120782445862435866478938747173134157139939886609445024737337773265122004941 3
0449600843184007188463149836820734043101946765328133425324305967455919056880354069778261 0
3975958300997458520792027696830112391960228973636908220967433149372586394273995865128301 5
5593045228702573569912114129397484571248740000246455550272630438526909144241610823738454 9
9317910748374967523586543285043142213534285268768000194626583474234347024441878880340815 5
9281928280944839979790920634154388105768112794024286496423645929859015106417339329908781 9
0251921061114025641223745643979128378494369005222470068384353754804565403745656524792770 3
1901135533561160495319726362039352982633654055056245839667558376405719514103007378428664 6
3212507133643869660760672915072023046556651108878336763783470573899380023554002159113637 5
5347922453142887678477325660960044296683495143833068929223268069448267517163735939570122 0
9218137570601492219025679827762150283860749832425855880588661258032068369664810440369773 9
4899435199838515763032245574018780878254767422461153074644961751053293671616515132655982 7
3789206916941964528634075793618577302671688452257201845245483178967685554512568295601521 3
0748388604293473386797295774207098085217055489
Signed Message:
3963440540339089934755311767689918893013567088112492722493882761554383671882229843096315 2
0916345363878427921945089307198988493529890529726390299391698354280549934746962106372113 8
9874544427777795886469201699503388108074199496770704059292021958003870822926715765216190 5
1809715575052170565861212344350086724362638738180090037655536062446701514823991542957443 4
2343670867059313458918912537780838379564304927491493572593311397647016885431153418926297 1
0195436769162820545498862033071796979264173769851206671215886374364370443897683920592631 7
3510869168526976447345201281580390667127598554415552385808465009431626543471914410374138 2
5259397410749643030270596662484013332388310420335012135088233808004513075279129814016001 2
5608370150391388536769673068691448533479197417144771240050498367791690269261952606815454 3
6307335421741726738414334949069009558212380446916691801662350324199250873174332443278445 7
3479851412193159194102464399861525759555212764299658722085435385964576325722123310156049 1
9571799682070910532068385991829239484372404344372934301477334366358269539425476850183014 2
```

9978792008497678007264339604753815233230743034989593739215472534256491395156605213296992559841580174929057775370858950301896373769562376151278423883877710287723457 80
ID: 575515818954178421565834089266231852771230696305
Subtracting 2 to 10
Returning sum of 8 to client

ID Verified: Pass
Signature Verified: Pass
Clear Message:
575515818954178421565834089266231852771230696305,3,0,65537,4460131306073789214833814333198805983992238120782445862435866478938747173134157139939886609445024737337773265122004941304496008431840071884631498368207340431019467653281334253243059674559190568803540697782610397595830099745852079202769683011239196022897363690822096743314937258639427399586512830155593045228702573569912114129397484571248740000246455550272630438526909144241610823738454993179107483749675235865432850431422135342852687680001946265834742343470244418788803408155928192828094483997979092063415438810576811279402428649642364592985901510641739329908781902519210611140256412237456439791283784943690052224700683843537548045654037456565247927703190113553356116049531972636203935298263365405505624583966755837640571951410300737842866463212507133643869660760672915072023046556651108878336763783470573899380023554002159113637553479224531428876784773256609600442966834951438330689292232680694482675171637359395701220921813757060149221902567982776215028386074983242585588058866125803206836966481044036977394899435199838515763032245574018780878254767422461153074644961751053293671616515132655982737892069169419645286340757936185773026716884522572018452454831789676855545125682956015213074838860429347338679729577420709808521705548 9
Signed Message:
1201404155730499141076223272781548182680445473222087124450314887661584144242641270484055142028986633982359704037024378038504144012184498813151839869304655551198129571344741618903382813016271571896147400894515918815878144010152757085768210486001344721318404333128069080728345497428451114176678945841209790854897472645123137432845565771730790563292868286036129803228806100488906391707411342392563044094738317529254014410066546842287944395461229159512186522751626403661401271707222337326915837594298997313157794938422054729267916507792820454172934378174846343330821383403874561592845420221204379517752364172388093914097319427765981509634858486425043948807326040277390256464306491632738076860145626789459763751694382363986732239271732420529246645872287492671654349270184497291275340248749368619804400938933028303537965422657481650683256530028885423731520557631908314982589313446870953808718273184794054261911613321714385228808711298072181020722951078850488961110196962660997213777748130477059390405299329109364663066753918624142993492897408044867099990403709564895858443986839969134535628098182857977542656437591099135676922700849311615318244078214244346133494760537403441753899416776246570741431195132564234651623528387113571756376199 45
ID: 575515818954178421565834089266231852771230696305
Getting sum...
Returning sum of 8 to client