Objective of this assignment:

- Develop and implement a simple application using TCP sockets. **The server must be implemented in a language other than Java.**

What you need to do

1. Implement a simple TCP Client-Server application

Objective:

For this programming assignment, you are advised to start from the code you developed for Programming Assignment 3. **For this assignment, the difference is that you must implement the calculator server using TCP in a language other than java.**

Part A: **Stream (TCP) Socket Programming**

The objective is to design a **Calculating Server (CS)**. This calculating server performs bitwise boolean and arithmetic computations requested by a client on 16-bit signed integers. Your server must offer the following operations: 1) addition (+), 2) subtraction (-), 3) multiplication (*), 4) division **AND** (/), 5) Shift Right (>>), 6) Shift Left (<<), and **not** (~).

A **client** **request** will have the following format:

Field	TML	Request ID	Op Code	Number Operands	Operand 1	*Operand 2*
Size (bytes)	1	1	1	1	2	2

Where

1) **TML** is the Total Message Length (in bytes) including TML. It is an integer representing the **total** number of bytes in the message.
2) **Request ID** is the request ID. This number is generated by the client to differentiate requests. You may use a variable randomly initialized and incremented each time a request is sent.
3) **Op Code** is a number specifying the desired operation following this table

Operation	+	-	*	/	>>	<<	~
OpCode	0	1	2	3	4	5	6

4) **Number Operands** is the number of operands: 2 for (+, -, *, /) and shifts. It is 1 for ~ (NOT).
5) **Operand 1:** this number is the first or unique operand for all operations.
6) **Operand 2:** this number is the second operand for operations (+, -, *, /, <<, >>). It is the number of bits to shift by for the shift operations. This operand does NOT exist for the ~ (NOT) operation.

Operands are sent in the **network byte order** (i.e., big endian).
Hint: create a class object *Request* like "Friend", but with the information needed for a request.
Below are two examples of requests

Request 1: suppose the Client requests to perform the operation 240 >> 4, i.e., shift the number 240 right by 4 bits (if this is the 7th request):

0x08	0x07	0x04	0x02	0x00	0xF0	0x00	0x04

Request 2: suppose the Client requests to perform the operation 240 – 160 (if this is the 9th request):

0x08	0x09	0x01	0x02	0x00	0xF0	0x00	0xA0

The **Server** will respond with a message with this format:

Total Message Length (TML)	Request ID	Error Code	Result
one byte	1 byte	1 byte	4 bytes

Where
1) **TML** is the Total Message Length (in bytes) including TML. It is an integer representing the **total** numbers of bytes in the message.
2) **Request ID** is the request ID. This number is the number that was sent as Request ID in the request sent by the client.
3) **Error Code** is **0** if the request was valid, and **127** if the request was invalid (Length not matching TML).
4) **Result** is the result of the requested operation.

In response to **Request 1** below

0x08	0x07	0x04	0x02	0x00	0xF0	0x00	0x04

the server will send back:

0x07	0x07	0x00	0x00	0x00	0x00	0x0F

In response to **Request 2**,

0x08	0x09	0x01	0x02	0x00	0xF0	0x00	0xA0

the server would send back:

0x07	0x09	0x00	0x00	0x00	0x00	0x50

a) **Repetitive Server**: Write a **stream Calculating Server (ServerTCP.xxx)** **in a language other than java**. This server must respond to requests as described above. The server must run on port (10010+*GID*) and could run on any machine on the Internet. **GID** is your group ID that I will assign you. The server must accept a command line of the form: *xxx ServerTCP portnumber* where *portnumber* is the port where the server should be working. For example, if your Group ID (GID) is 13 then your server must listen on Port # 10023.

b) Write a **stream client** (**ClientTCP.java**) in java:
   i. Accepts a command line of the form: *java **ClientTCP** servername PortNumber* where *servername* is the server name and *PortNumber* is the port number of the server. Your program must prompt the user to ask for an *Opcode,* an *Operand1* and if needed an *Operand2* where *OpCode* is the opcode of the requested operation (See the opcode table). *Operand1 and Operand2* (if applicable) are the operands. For each entry from the user, your program must perform the following operations:
   ii. form a message as described above
   iii. send the message to the server and wait for a response
   iv. print all the message one byte at a time in hexadecimal (for debugging purpose)
   v. print out the response of the server in a manner convenient for a typical Facebook user: the request ID and the response
   vi. print out the round trip time (time between the transmission of the request and the reception of the response)
   vii. prompt the user for a new request.

How to get started?

1) Start with the code you developed for Programming Assignment 3. If your program assignment 3 did not work, you are allowed to use the working client from another group

Report

- Write a report. The report should not exceed half a page.
- Your report state: whether your programs work or not (this must be just ONE sentence). If your program does not work, explain the obstacles encoutered.

What you need to turn in:
- Electronic copy of each source program separately (standalone). **In addition**, put all the source programs in a folder that you name with your group ID. Zip the folder and submit it **TOO**.
- Electronic copy of the report (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.

Grading

1) TCP server is worth 80% if it works well: communicates with YOUR client.
2) TCP server is worth 20% extra if it works well with a working client from any of your classmates.