Jordan University of Science & Technology
Department of Network Engineering and Security
NES416- Network Programming
Programming Assignment 2 (CLO2, 3)

Due Date: see course E-Learning site

Purpose: Design and develop a simple iterative TCP client server programs
**Note: this assignment is done in a group of two students at most. Record the name and ID of team members as a comment at the beginning of your source code*

Description:

Use TCP socket programming in C to implement a pair of **iterative** client and server programs. The server waits for client's request which represents operands for a mathematical operation, evaluates the operation, and sends the results back to the client. The client asks the user to select the operation from a menu, then enters the operands, sends them to the server for evaluation. Once the reply from the servers arrives, the client displays the result to the user and asks for another operation by redisplaying the menu, and so on. Note that the connection between the client and server should stay open, so that the client can repeat the operation again until the user selects exit option from the menu, at which time the server exits also

At the very minimum, the server should be able to handle addition, multiplication, subtraction, and division operations on two 16-bit signed integer operands. The menu displayed as follows:

```
1. Add
2. Mul
3. Sub
4. Div
5. Exit
```

Note that, you need to send the operands and the operation to be performed from the client together in a structure. In addition, you need to consider the result also as member in this structure. It is up to you to choose how to encode the operation in this structure. The server computes the required operation, and uses the same structure to insert the results and sends it back to the client.

The client program should use command line arguments to read the IP and Port numbers of the server. Don't use bind() system call at the client to assign the local protocol address.

Furthermore, the server should use command line arguments to indicate the port it will listen to. Once the client prints the received replyfrom the server, it also prints the IP and port number of the server. Also, once the server receives the clients requests, it is locally displayed together with the client's IP and port number

Output Sample:

For example, running the code should produce something similar to this output:
Client_side> *HW2 <server's IP> <server's port#>*

```
1. Add
2. Mul
3. Sub
4. Div
```

5. Exit

please enter your choice:
1 (input from a user)
Enter two 16-bit signed operands
123
-111
Client_side > 123 + (-111) = 12 (***server IP , Port#***) <where 22 is the answer received >
Client_side >

1. Add
2. Mul
3. Sub
4. Div
5. Exit

please enter your choice:
5 (input from a user)
Client_side > exiting (→then the client side exits, and also the server side)

Server_side > waiting for client messages.
Sever_side > received "123 , -111" from the client (**IP address, port #**)
Server_side > Sending " 22 " to the client
Server_side > waiting for client messages
Server_side > received "exit" from the client (**IP address, port #**)
Server_side > exiting (→ then the server exits)

Hints:

- ☐ DO NOT use the header file "unp.h" from the book
- ☐ Each student should use a port number for the server such that is larger than 45000
- ☐ Your program for client needs to take two arguments that specify the IP address of the server and the port that it is trying to connect to. Your program for server needs to take an argument that specifies the port that it is listening to (the same one provided to the client)
- ☐ Don't use bind at the client's side
- ☐ Ask questions as early as possible.
- ☐ Submit your source code for both the client and the server, their compilation, and some running sample of your code as one zipped file whose name is your student ID number
- ☐ Your programs should be compiled and run without any single error or warning.
- ☐ **Comment and error-check you code**