CMSC 417 Computer Networks **Spring 2010**

Programming Assignment 0

Assigned: January 29, 2010 *Due: February 5, 11:59:59 PM.*

You should use the CSIC Linux cluster systems and obtain your account information from the TA.

1 Background

A Pythagorean triple is a triple of **positive integers** a, b, and c such that a right triangle exists with legs a, b and hypotenuse c. By the Pythagorean theorem, this is equivalent to finding **positive integers** a, b and c satisfying:

$$a^2 + b^2 = c^2 \tag{1}$$

The most well known Pythagorean triple is $(a, b, c) = (3, 4, 5)$.

2 Description

In this assignment you will develop a **TCP** client. The client's goal is to communicate with a server program (that we have written) and answer correctly the challenge that it receives from the server. For each client request, the server generates a possible Pythagorean triple and the client has to decide if it is a valid triple or not. The client and server terminate the conversation by exchanging a sequence of goodbye messages. Despite the fact that the protocol is trivial, the assignment will get you started with sockets and network programming.

3 The Protocol

The server runs on the machine at `SERVER_IP` and listens for requests on a TCP socket bound to port `SERVER_PORT`. All constants are defined in the header file provided for you. This protocol has four types of messages: `CONNECT`, `CHALLENGE`, `TRIPLE_REPLY`, `CONFIRM_EXIT`. Each message is an ASCII string and consists of multiple fields seperated by a '#' (number sign). All strings must end with the following character sequence `"\n\n\r"`.

The protocol outline is given in Figure 1. The client initiates the protocol by sending a `CONNECT` message to the server. The server replies with a `CHALLENGE` message. The client then sends a `TRIPLE_REPLY` message, and the server terminates the conversation with a `CONFIRM_EXIT` message. The session is successful if and only if all of these messages are sent and received correctly and the client's answer in the `TRIPLE_REPLY` is correct.

The details of each message are as follows:

- `CONNECT` (Client → Server)

  The `CONNECT` message has 3 fields EXACTLY in the following order

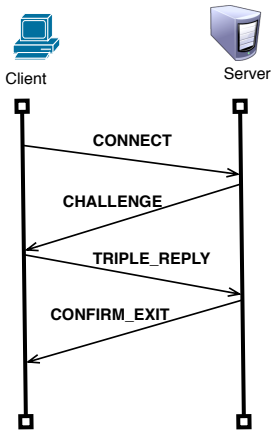

Figure 1: Assignment's Protocol

  - **Message Type**
    The message type MUST be "0" to indicate a message type CONNECT.
  - **Magic String**
    It MUST set to be `MAGIG_STRING` which is a constant ("cmsc417spring2010") defined in the header file. If your message does not contain this magic string it will be ignored.
  - **University ID**
    This field is your cluster University ID.

  An example `CONNECT` message:
  `"0#cmsc417spring2010#1107402504\n\n\r"`

- `CHALLENGE` (Server → Client)

  The `CHALLENGE` message has 5 fields in the following order:

  - **Message Type**
    Must be set to "1".
  - **Magic String**
    Same as above.
  - **A**
    An integer generated by the server (represented in ASCII).
  - **B**
    An integer generated by the server (represented in ASCII).
  - **C**
    An integer generated by the server (represented in ASCII).

  An example `CHALLENGE` message would be:
  `"1#cmsc417spring2010#3#4#5\n\n\r"`

- `TRIPLE_REPLY` (Client → Server)

  The `TRIPLE_REPLY` message has 7 fields in the following order:

  - **Message Type**
    Must be set to "2".

- **Magic String**
  The same as above.
- **University ID**
  Same as above.
- **A**
  The same as above
- **B**
  The same as above
- **C**
  The same as above
- **RESULT**
  It MUST set to be 1 if the integers in the `CHALLENGE` are a valid Pythagorian triple and 0 otherwise. Messages with something other than 0 or 1 will be ignored.

An example `TRIPLE_REPLY` message for a valid triple would be:

`"2#cmsc417spring2010#110740254#3#4#5#1\n\n\r"`

- `CONFIRM_EXIT` (Server → Client)

  The `CONFIRM_EXIT` message has 2 fields EXACTLY in the following order:

  - **Message Type**
    The message type MUST be "3 " to indicate a message type `CONFIRM_EXIT`.
  - **Magic String**
    The same as above.

  An example `CONFIRM_EXIT` message would be: `"3#cmsc417spring2010\n\n\r"`

4 The client program

The command line syntax for the client is given below. The client program takes command line argument corresponding to university id. The SERVER_IP as well as the SERVER_PORT are defined in common.h.

```
client <university id>
```

5 Requirements

You may test your client code with our server as many times as you like. Your client should conform to the protocol described above, or otherwise the server will terminate the connection silently.

Your client program must verify the validity of messages by checking the magic string and message type fields in `CHALLENGE` and `CONFIRM_EXIT` messages. If a received message is not as expected, such as an incorrect magic string or unexpected message type, assert an error and terminate your program.

Your code must be **-Wall** clean on gcc. Do not ask the TA for help on (or post to the forum) code that is not **-Wall** clean unless getting rid of the warning is what the problem is in the first place.

6 Hints

At the `CHALLENGE` message from the server lists a,b,c in such a way that always will satisfy the following relationship: $a < b < c$. Thus, you don't have to worry about which of a, b, c would be the legs of the triangle and which the hypotenuse.