CPSC 3600 Homework #1
Spring 2015

Assigned Jan. 12, 2015
Due Jan. 30, 2015, 5pm

To be completed individually

You are to create a client server program that will have you develop your programming abilities. We will call the application *Value Guesser*. Use the UDP echo server as an inspiration for your submission, but you will need to edit this code to complete the assignment.

The server side comes up with an unknown positive integer value. The client's job is to guess that integer value. Once the client has a guess, it sends a message to the server asking if the value is correct. The server replies with a message of 0, 1, or 2. **Zero** is returned if the guess is correct, **1** if the guess is too high, and **2** if the guess is too low. At the server, if the value is guessed, the server randomly comes up with a new value and loops to play the game again. At the client, if it successfully guesses the value it terminates displaying an appropriate message to the user.

If each query takes 2 milliseconds, it would take less than 5 minutes to test 1 million values. We describe the client and server programs in more detail.

Client program (you must call the program valueGuesser) syntax:

valueGuesser <serverName> <serverPort>

serverName: The name (domain name, IP address in dotted name format) of the host that is running the server side.

serverPort : the 16 bit port number the server is using. You should generally run the server with a port number in the 5K to 10K range.

The client runs until either it succeeds to guess the server's value or until a CTRL-C is issued. Upon termination, the program must display to standard out the number of attempts and the total time (in seconds to microsecond precision) the program ran and the guessed value.

Client output format is below:

Attempts<\tab>Time<\tab>Value

An example of client output:

27 12383.092 542

The client must communicate with the server using a UDP socket. The client loops forever in the following loop:

```
while (true)
        nextValue = selectNextValue()
        returnCode = tryValue(nextValue)
        if returnCode == 0, print completion message and break;
        if returnCode == 1, continue looping
        if returnCode == 2, continue looping
        if returnCode == ERROR_TIMEOUT, continue looping
end while
```

The client must handle the case if a message to the server is dropped. The client must initiate a timeout each time it sends a message to the server. If the timeout pops, the client simply continues to the next iteration. A reasonable value for the timeout duration is 1 second (you might need to increase this in the event other students are using the same machine and the response time is very slow).

Server program (you must call the program valueServer) syntax:

valueServer serverPort initialValue

serverPort : the 16 bit port number the server should use. You should generally run the server with a port number in the 5K to 10K range.

initialValye: This parameter is optional (rather…you must support it. Using it when invoking your server is optional). If the parameter is not specified, the server will generate an initial value randomly.

The default behavior for the generator is to select any possible value in the range (0, 10E9) with equal likelihood. If the parameter is specified, after the value is guessed, the server will generate the next value using the random algorithm.

The server program runs forever or until a CTRL-C is issued. Upon termination, the program should display the total number of messages received and the total number of times a client correctly guessed the value.

Server output format is below:

#Messages<\tab>#Clients<\tab>IPs

An example of the output is below (note the commas between IP addresses!):

100 2 12.23.1.32, 12.23.1.32, 12.23.1.35,

The server should also display a list of all clients (specifically, their IP address) that attempted to guess the value. The server should run a loop similar to the following:

```
currentValue = generateValue()
while (true)
        clientsValue= receiveNextClientValue()
        returnCode = testClientsValye(clientsValue)
        if (returnCode == 0)
                currentValue = generateValue ()
        return returnCode reply to client
end while
```

Submission: Please submit a single tar.gz package (youruserid-hw1.tar.gz) **without including subdirectory** via handin (http://handin.cs.clemson.edu):

> tar cvzf youruserid-hw1.tar.gz README Makefile <list of source files>

> handin youruserid-hw1.tar.gz

Your tar.gz must include the following:

- A Makefile that works. If your code does not correctly build and run after I issue a 'make clean; make depend; make' you will lose LOTS of points!!!
- Well-documented source code. Each file (.h and .c file) should have a file header that describes what is contained in the file and lists the routines. Each routine should have a routine header that describes the parameters and gives a 1-2 sentence overview of what the routine is doing.
- A readme.txt file that explains the over all structure of your design and implementation, how to make your program, how to run the client and server. To get a feel for how much detail….use this as a guide: If I give your package to a CS student who is not in our class, he/she should be able to understand how to extend and run your code with minimal effort (i.e., without having to study the source code).

WARNING: All programming assignments will be designed to facilitate automated testing. There will be HUGE penalties for programs that do nothing more that either do not build correctly (all submissions must build with a 'make') or that core dump on execution. To be eligible for any partial credit, you must have some portion of the program working correctly.

Once again, you are to complete this assignment on your own.

We will abide by the University academic integrity policy
(http://www.clemson.edu/academics/integrity/)