Introduction
The purpose of this assignment is to give you some experience with the construction of a network client using the TCP protocol. You will not construct the server, as it has already been written and tested by the instructor.

Recall that virtually all TCP client programs will minimally do the following things:

1. Create an appropriate socket.
2. Connect that socket to the server at the appropriate IP and port address.
3. Communicate with the server (send messages to it and receive messages from it).
4. Terminate the connection.

There will usually be some work done during step 3 to determine what to do with the data in the messages received from the server and to determine what messages to send to the server.

In this assignment, your program will play a guessing game with the server. In particular, the server will pick an integer in the range 0 to 999,999,999 (inclusive) and your program will try to guess the number picked by the server. Each time your program makes a guess (that is, picks a number and sends it to the server), the server will respond by indicating if the guess was too high, was too low, or was perfect.

Details
Before the guessing game begins, but after the connection to the server has been established, your client program must send your student ID to the server. This message will contain exactly 9 data bytes: 8 bytes containing the ASCII character codes for the digits of your student ID, and an end of line character (\n). If the ID message is accepted, the server will respond with a welcoming message containing no more than 100 characters terminated with an end of line character. Your program should display this message on the standard output. It will look something like this:

Welcome, Alan M. Turing

Of course the ID message will have your name instead of the famous name shown here.

It is very important to realize that the message should not be terminated by a null byte (that is, a byte containing the value 0x00). In particular, if you are using a C-style character string, the C library functions typically require a null byte to mark the end of the string. The message itself – that is, the sequence of bytes sent as data – should not include that null byte (at least for this program).

If your student ID is not recognized by the server, or if your message is improperly constructed, the server will usually send an error message and terminate the connection with your client program.

But let’s assume you send your student ID successfully and the server accepts it. Now the guessing game begins. To make a guess, prepare a message containing between 1 and 9 bytes consisting of the ASCII codes for the decimal digits in your guess, followed immediately by an end of line character, and send that message to the server. For example, suppose your first guess was 32. In that case, your message might contain three bytes with the values 0x33, 0x32, and 0x0a – the ASCII codes for the
characters '3', '2', and end of line. Leading zeroes (that is, '0') may be used, if desired, but the entire message must not contain more than 10 total bytes, including the end of line character. It must not contain any spaces or other characters (except for the end of line, which must be the last character). If the message is improperly formatted, the server will respond by sending your client an appropriate error message and terminating the connection.

If the message containing your guess is acceptable, the server will respond with one of three messages:
"HIGH\n", "LOW\n", or "CORRECT!\n". The first response (HIGH) will be sent if the number your program guessed is larger than the number the server picked. Obviously the second response (LOW) will be sent if your program’s guess is too small, and the final response is sent if your program picked the correct number.

In event your program didn’t guess the correct number, it should try again – that is, pick a different number, prepare the appropriate message, and send it off to the server. If you did pick the right number, then your client should display the correct number (on the standard output), close the connection with the server and terminate.

The server will only wait up to 500 milliseconds (that is, half a second) to receive each message from your client, so you probably can’t read guesses from the keyboard and send them off to the server – unless you’re a very fast typist! Also, the server gets bored easily, and will terminate the connection if you make 35 incorrect guesses. So your program had better be smart about how to make those guesses!

Notes and Restrictions
Your solution may be written in the C, C++, Java, or Python programming language. It must not use or need any command line arguments or aby data read from the standard input.

The server for this assignment will be running on the host cs2.ist.unomaha.edu, port 83. You will likely need to run all your test programs on loki (or another university system), since network administrators are (perhaps) paranoid about allowing “outside” connections to most university systems (except a few systems like loki, and connections to web servers). All connections to the server, and much of the communication with clients will be logged.

A library of functions for use with C and C++ programs is provided in the /home/stanw/csci3550/lib directory. You may use these to simplify the construction of your solution. A brief document describing the functions is also located in that directory (or soon will be placed there).

Evaluation
How will programs be evaluated? 50% of the grade will be based on a reasonable program (with appropriate internal documentation) that can at least connect to the server. Another 15% will be given if the program can “log on” – that is, it can send your student ID and receive the server’s response. If that works, then it should be easy to get another 10% if you can send at least one valid guess. The final 25% of the grade will be based on your program’s ability to guess the correct number and then close the connection with the server.

Requirements
You must write (and test) a program that functions as a simple number guessing client as just described. Your solution should ideally be contained in a single file of source code; multiple files will be accepted but only if there is appropriate rationale for using multiple files; a C/C++ solution using the library
functions provided in the /home/stanw/csci3550 directory should not include the library. The solution must be submitted by Thursday, April 5, 2018 by 11:59 PM.

The source code file (or files) for your solution should be placed in a directory named csci3550-181-prog2 immediately below your home directory. For example, if your username was brenfro, your home directory would probably be /home/brenfro and your solution’s source code would be placed in the directory /home/brenfro/csci3550-181-prog2.

As always, please contact the instructor if you have questions, and periodically check the class web site for any additions or corrections to this assignment.