CSE 3100 Systems Programming	Fall 2023
Lab #9	

Welcome to Lab 9! A Makefile is provided. Submit your work to gradescope.

Part 1. (100 points) Guess My Number - Socket. In this lab, we implement another version of **Guess My Number** game, using the Internet sockets! After completing the assignment, you can set up the online game at home and invite your parents, siblings, or guests to play! (and show off how much you know about binary search!)

The game is the same as described in Labs 6 and 8. In this problem, it is more natural to call the two people in the game S and C. S is on the server side and knows the random number. C is a client that connects to the server and uses binary search to guess the number.

The server listens on TCP port 3119. Clients connect to the port when they want to play the game. Once a connection is established, the server creates a thread S to play game with the client C. Multiple clients can play the game simultaneously. The protocol is as follows.

1. After a client is connected, S first tells C the largest possible value.
2. C makes a guess (picking an integer in the range) and tells S the guess.
3. S checks if the guess is correct and informs C the result. The result is 0 if the guess is correct, 1 if the guess is smaller, or -1 if the guess is larger. If the guess is correct, S also sends a final message to C and then exits.
4. If the guess is not correct, C adjusts the range, based on the feedback from S, and goes to Step 2.
5. If the guess is correct, C prints the final message to `stdout` and exits.

The communications between S and C are line based. Each line must end with a new line character. Integers are converted to strings of decimal digits (with an optional negative sign) before being written to the socket. Converting integers between native binary format and decimal strings can be done by functions like `snprintf()`, `sscanf()`, `atoi()`, or `strtol()` (BTW, do NOT use `sprintf()`!). Read the manual to learn how to use these functions. For example, the following function call converts integer `n` to a string in `buf` of `sz` bytes. Make sure `buf` is large enough. Although we do not have a lot of surprises when printing an integer, it is a good practice to check the return value!

```
rv = snprintf(buf, sz, "%d\n", n);
// check rv
```

The protocol is straightforward. One line is transmitted for each message. Note that S sends a single line at the end when the guess is correct: `"0"` (without newline) followed by the final message. If we use `recv_lines()`, the client will receive both the zero and the final message in one call. Once the client receives the final message, it will need to print the received 0 on a separate line (just like when the client prints out the other responses, 1 and -1), and print the final message on the second line.

Search for `TODO` in the starter code (`server.c` and `client.c`) to find the places where code is to be completed. The steps are similar to what we have done in previous labs.

We should work on the server first. We can revise the client from the demo program `s2.toupper` to have a program to talk to the server manually. We can also use commands like `nc`, if they are available, to test the server.

```
$ nc localhost 3119
$ seq 1000 | nc localhost 3119
```

The output of the client is similar but different from the program in previous labs. The client prints a "connecting to" message once the connection is made. Also, it prints the responses from the server. A sample session is as follows.

```
$ ./client localhost
client: connecting to 127.0.0.1
1000
My guess: 500
-1
My guess: 250
1
My guess: 375
-1
My guess: 312
-1
My guess: 281
1
My guess: 296
0
It took you 6 attempt(s) to guess the number 296.
```

This will be the last guess my number program. We promise!

Happy coding!