CSE 3100 Systems Programming	Fall 2024
Lab #8	

Welcome to Lab 8! Read each question carefully before beginning work; A Makefile is provided. Submit your work to gradescope.

Part 1. (100 points) Guess My Number - MT. In this lab, we implement the **Guess My Number** game again, using threads! The game is the same as described in Lab 6, and we still call the two people in the game C and P. C generates a random number and P uses a binary search to guess the number. In the multithreading version, C and P are threads. Although they share the memory space, P is honest and does not peek at the answer. No cheating!
In this problem, thread C and thread P use a shared structure (`thread_arg_t`) to exchange information. Of course, they need a mutex and condition variables to coordinate. In the starter code, the structure already has one mutex and two condition variables. The status field indicates the status of the shared data.
Let us recap the protocol between C and P.

1. C first tells P the largest possible value.
2. P makes a guess (picking an integer in the range) and tells C the guess.
3. C checks if the guess is correct and informs P of 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, C also copies a final message to the shared structure and then exits.
4. If the guess is not correct, P adjusts the range, based on the feedback from C, and goes to Step 2.
5. If the guess is correct, P prints the final message to `stdout` and exits.

Thread P uses a binary search to minimize the number of the guesses. `guess_my_number()` in the starter code demonstrates the process.

The tasks in this problem are to complete the starter code so the main thread creates two threads, C and P, that play the game by exchanging information through the shared structure `thread_arg_t`.

Search for `TODO` in the starter code to find the places where code is to be completed. The detailed steps are described in the comments. Below are some requirements and suggestions on the implementation.

- Thread C uses local variable `gmn` to maintain the game state and calls provided functions to operate on it. The thread does not access fields in the structure directly.
- There are many ways to synchronize the two threads in this problem. Let us use one mutex and two condition variables. We want to practice for more complicated programs! After solving the problem with two condition variables, you can experiment with other methods.

The program `guess-my-number` takes optional arguments from the command line. An argument can be one of the following.

- A positive integer. It will be used as the seed for generating pseudorandom numbers. If no seed is specified on the command line, a default value is used.
- `demo`. Call the demo function and exit. It helps us to test our implementation using two threads.

The output of the program should be the same as in the demo mode, as well as the output from the program in Lab 6.

```
$ ./guess-my-number demo 12345
My guess: 500
My guess: 750
My guess: 875
My guess: 938
My guess: 969
My guess: 985
My guess: 993
My guess: 997
My guess: 999
My guess: 1000
It took you 10 attempt(s) to guess the number 1000.
```