CS271 (Spring 2018) Lab 8

Due: Apr 1, 2018 (Sunday) before midnight at 11:59pm

Submit your C programs in a zip file to CANVAS

In this lab, we write C programs that play the mastermind game. For instructions about mastermind, you can read it from the web page at Wikipedia. One person is playing the role of "code maker" and another person is the "code breaker". For this lab, you can use the programs codeMakerSimple.c and codeBreakerSimple.c as a starting base. You can freely use it, modify it, and submit your modification of it as your solution to this lab.

The code maker program set the codes by guessing. The code breaker program tries to break the codes also by guessing. Ideally, the code breaker should consider the history of guesses and code maker's replies in determining the next guess. However, the given code breaker program is very simple-minded. It relies on simple guessing without taking into account of the information available.

In this lab, you are asked improve both the code-maker and code-breaker programs.

Starting the System

The "mastermind" script is the game controller, and is a Tcl script. The script is a text file. You can look at the codes for the script.

NOTE: After downloading the "mastermind" script, be sure to set the *execute* permission bit "on" for the script file. Otherwise, it won't be runnable.

To compile the sample code-maker and code-breaker programs, type:

```
gcc codeMakerSimple.c -o codeMakerSimple

gcc codeBreakerSimple.c -o codeBreakerSimple
```

Then you can run the code breaker program against the code maker program as:

```
tclsh mastermind -codemaker codeMakerSimple -codebreaker codeBreakerSimple
```

Once you develop your own code maker program and code breaker program,

say with the executables called "codeMaker" and "codeBreaker" respectively, replace the run command as:

```
tclsh mastermind -codemaker codeMaker -codebreaker codeBreaker
```

If you get an error such as "command not found", then "tclsh" is not being found in your search path or on your machine. Also, be sure that the script file is executable.

Protocols between Game Controller, code Maker and Code Breaker

The code maker program communicates with the game controller by reading from and writing to "stdin" and "stdout" respectively. In addition, the code maker program maintains a log file (`codeMaker.log`) for all the communications with the game controller. Similarly, the code breaker program communicates with the game controller by reading from and writing to "stdin" and "stdout" respectively. Also, the code breaker program maintains a log file (`codeBreaker.log`) for all the communications with the game controller. The code maker reads from the game controller a line of 4 integers denoting the guess from the code breaker. Note that we represent the six colors by integers 0, 1, 2, 3, 4 and 5. The code maker's reply is sent to stdout as two integers that denote the number of pegs that are correct in both color and positions and the number of pegs that are correct in color only, but not in positions. Next, the game controller forward the code maker's reply to the code breaker through stdin. The code breaker's next guess is sent to the stdout, which is then picked up by the game controller and forwarded to the game maker.

Task 1

The code breaker's program (`codeBreakerSimple.c`) is very simple-minded. You need to improve the codes to include the following logic:

- The code breaker should NOT submit a guess that is the same as a previous guess. That is, it should maintain a history of the guesses made so far. If a guess is not "new", it should generate another guess instead.

- The code breaker should check to see if the new guess is consistent with all the previous replies from the code maker. For example, if the reply to the guess

  ```
  2 4 3 1
  ```

  is (0, 0). In that case, it can be concluded that the correct guess should not include colors 1, 2, 3 or 4. Therefore, the new guess should not be, say,

```
0 5 0 2
```

as it is wrong to include color 2.

- You can include other logic to make your code breaker program more "smart".

Task 2

The code maker's program (`codeMakerSimple.c`) can change the "answer" after the game has started as long as the "new answer" is consistent with all the previous replies that the code maker program made in response to the code breaker's guesses. For example, suppose initially the code maker's answer is

```
1 0 3 0
```

For the first attempt, the code breaker guesses

```
0 2 1 3
```

As expected, the code maker's reply is (0,3). Next, suppose the code breaker guesses

```
1 0 3 0
```

The code maker realizes that the guess is correct. But, the code maker program also knows that the guess is really lucky. Instead of admitting the guess as correct, the code maker changes the answer to, say,

```
4 0 2 1
```

The new answer is still consistent with the reply (0,3) that the code maker gave for the first guess of 0, 2, 1, 3. With the new answer, the code maker answers the second guess by sending a reply of (1,1)

- You have to incorporate some kind of "mind-change" logic into the code maker program.

Extra Credit Problem

We will run a competition among the students' code maker and code breaker programs.

Competition rules:

- Suppose we have two students A and B. We run A's code maker program against B's code breaker's program. Suppose B needs 7 guesses to break

the code. On the other hand, suppose A's code breaker needs 4 guesses to break the code given by B's code maker program. In that case, we say that A wins.
- The student's programs with the most wins is the champion.
- If there are ties for the most wins, we will have a run off between the two students' programs.

Reward: The student's programs with most wins will earn the credits of the equivalent of two labs. The student's programs with the second most wins will earn the credits of the equivalent of one lab.