CSC3150 Assignment 2

Homework Requirements

Environment

- **WARNING!!!** Before starting on this assignment, make sure you have set up your VM properly. We would test all students' homework using the following environment. You can type the following command in terminal on your VM to see if your configuration matches the test environment. If not, you are still good to go, but please try to test your program with the following environment for at least once. Because you may be able to run your program on your environment, but not on TAs' environment, causng inconvenience or even grade deduction.

  If you follow the tutorials then your VM setting should be fine, though verify your environment again is recommended.

- **Linux Version**

- Ubuntu 20.04, but others are ok. You can use the following command to get it.

```
main@ubuntu:/$ cat /etc/issue
Ubuntu 20.04
```

- **Linux Kernel Version**

  4.4 or 5.10.x is ok (Test Environment). You can use the following command to get it.

```
main@ubuntu:/$ uname -r
5.10 (or 4.4)
```

- **GCC Version**

- 4.9 above. Use "gcc --version" to get it.

```
main@ubuntu:/$ gcc --version
gcc (Ubuntu 5.4.0-6ubuntu1~16.04.10) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Submission

- **Due on: 23:59, 24 Oct 2022**
- Please note that, teaching assistants may ask you to explain the meaning of your program, to ensure that the codes are indeed written by yourself. Please also note that we would check whether your program is too similar to your fellow students' code using plagiarism detectors.
- Violation against the format requirements will lead to grade deduction.

Here is the format guide. The project structure is illustrated as below. You can also use `ls` command to check if your structure is fine. Structure mismatch would cause grade deduction.

Please mark whether you have finished bonus in your report.

```
main@ubuntu:~/Desktop/Assignment_2_<student_id>$ ls

Report.pdf 3150-p2-bonus-main/  source/

Two directories and one pdf, the "source/" directory is for the  game “Frog
crosses river” and "3150-p2-bonus-main/" directory is for the bonus task.
```

Please compress all files in the file structure root folder into a single zip file and **name it using your student id as the code showing below and above, for example, Assignment_2_118010001.zip.** The report should be submitted in the format of **pdf**, together with your source code. Format mismatch would cause grade deduction. Here is the sample step for compress your code.

```
main@ubuntu:~/Desktop$
 zip -q -r Assignment_2_<student_id>.zip Assignment_2_<student_id>

main@ubuntu:~/Desktop$ ls
Assignment_2_<student_id>                Assignment_2_<student_id>.zip
```

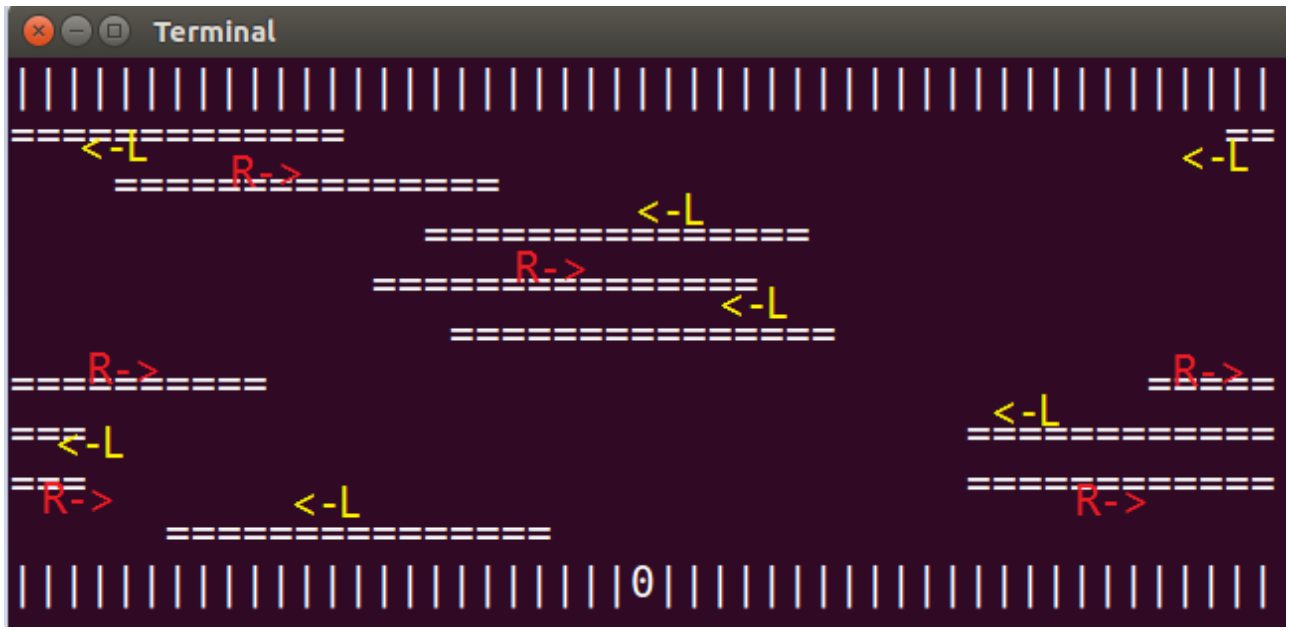
Task Description

In the **""source/"" directory** of Assignment 2, you are required to complete the multithread program to implement the game “Frog crosses river”.

Game rules:

- A river has logs floating on it, and a frog must cross the river by jumping on the logs as they pass by.
- Objects:
    - Log: ==========
    - Frog: 0
    - River bank: ||||||||||||||||||||

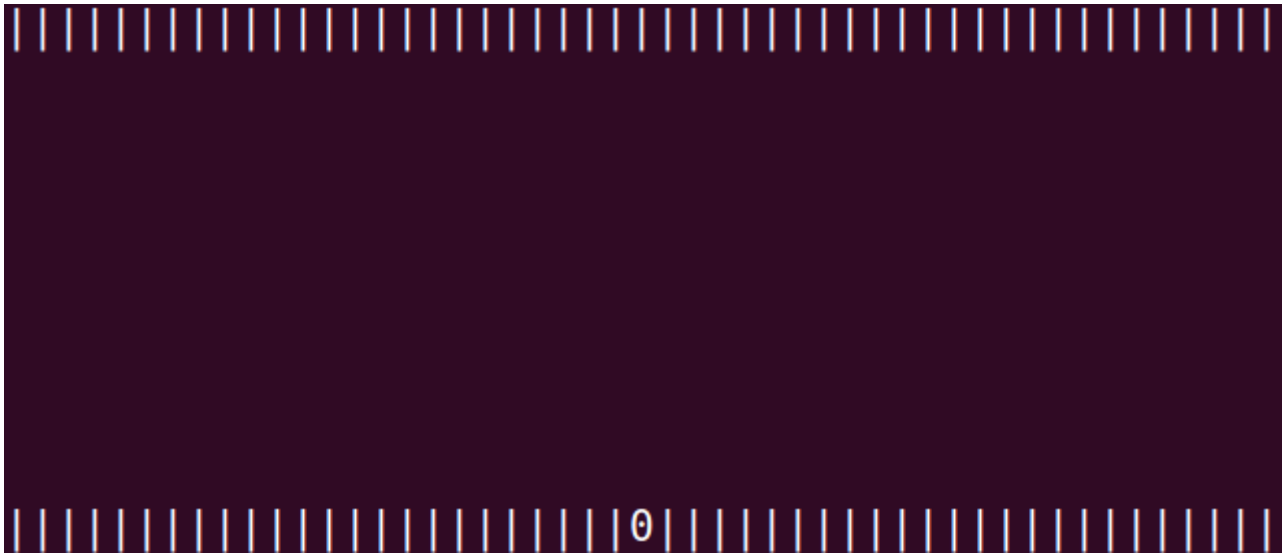
- When the game starts, the frog stands in the middle of bottom bank of river. The user can control the frog jumps by keyboards. The logs will move from left to right or right to left staggerly. Please take care of the edge cases.
    - W: UP
    - S: Down
    - A: Left
    - D: Right
    - Q: Quit the game



- You will win if the frog jumps to the other bank of river successfully.
- You will lose if the frog lands in the river, or the log reaches the left/right side of the river but the frog still on it.
- **Please note that this is a game.** If the player suffers from extereme difficulty, high latency, and other unexpected situations for a game, your grade will suffer as well.

Function Requirements (90 points):

- To run the template, you will see the frog stands in the middle at bottom bank of river. There are 9 blank rows which means the river. Compile the program to see the static output. (5 points)



- You should complete the function named “logs_move” to let the logs can move staggerly from left to right or from right to left. (20 points)
- You should create pthread and use mutex lock for logs and frog movement control. (30 points)
- “kbhit” function is provided for keyboard capture. You should complete the jump rules for keyboard actions. And the frog’s position should be updated when keyboard hits. (15 points)
- When the logs are moving, the program should be able to judge the game status (win, lost or quit). Print out the message for user whether he/she wins or lost the game. (15 points)
- If the user quits the game, print the message. (5 points)

Demo Output

- Demo output for user wins the game

```
You win the game!!
main@ubuntu:~/Desktop/Assignment_2_example/source$
```

- Demo output for user loses the game:

```
You lose the game!!
main@ubuntu:~/Desktop/Assignment_2_example/source$
```

- Demo output for user quits the game:

```
You exit the game.
main@ubuntu:~/Desktop/Assignment_2_example/source$
```

Bonus Task (10 pionts)

Attention: The bonus task has no relationship with the the above game "Frog crosses river".

In the **"3150-p2-bonus-main/" directory** of this bonus task, you have to implement correctly two important functions of a thread pool. The two functions in **async.c** are as follows:

1. `void async_init(int num_threads)`(3 points)
2. `void async_run(void (*handler)(int), int args)` (7 points).

```
7   void async_init(int num_threads) {
8       return;
9       /** TODO: create num_threads threads and initialize the
        thread pool **/
10  }
11
12  void async_run(void (*hanlder)(int), int args) {
13      hanlder(args);
14      /** TODO: rewrite it to support thread pool **/
15  }
```

- Implement in **async.c and async.h**: `void async_init(int num_threads)` and `void async_run(void (*handler)(int), int args)`
- You can use list data structure in utlist.h, for example: `DL_APPEND(my_queue->head, my_item);` (adding to queue end) and `DL_DELETE(my_queue->head, my_queue->head);`(popping from queue head)
- When no jobs are coming, your threads created in `async_init` have to go to sleep and is not allowed to do busy waiting like `while(1){sleep(any);}`, and when jobs are coming a sleeping thread in your thread pool **must** wake up immediately (that is, no `sleep()` call is allowed).
- async_run should be asynchronous without further call to pthread_create, that is it should return immediately before the job is handled (in the code we give you, async_run runs synchronously, so you need to rewrite the function)

After finishing these two function and the implementation of thread pool, you can make with the provided **Makefile** and a http server will be run based on the thread pool you implement so that you can see whether your implementation works or not.

Please see the directory **3150-p2-bonus-main** and read **README.md** in detail to do this bonus task.

What is a thread pool? Please see: https://en.wikipedia.org/wiki/Thread_pool.

Report (10 points)

Write a report for your assignment, which should include main information as below:

- Your name and student id.
- How did you design your program?
- The environment of running your program. (E.g., version of OS and kernel)
- The steps to execute your program.
- Screenshot of your program output.
- What did you learn from the tasks?

Grading rules

Here is a sample grading scheme. Different from the points specified above, this is the general guide when TA's grading.

Completion	**Marks**
Bonus	10 points
Report	10 points
Completed with good quality	80 ~ 90
Completed accurately	80 +
Fully Submitted (compile successfully)	60 +
Partial submitted	0 ~ 60
No submission	0
Late submission	Not allowed