1. Implementation of the program

   The main idea of the program is to implement the "frog cross river" game using multithread method.

   a) main( ):

      In the main( ) function, the program will first set the "flag" to 0, which means the game is continuing normally (Notice :
      flag = 0 for normal status, 1 for lose, 2 for win, 4 for quit). Then it will initialize the map of the game. And it will use logsInitialization( ) function to initialize the logs of the map. Then it will create two threads to manage the logs' and the frog's movements respectively using pthread_create( ) method. These two threads will running the logs_move( ) function with different parameter. The program will use pthread_join( ) method to wait for these two processes to terminate. After that, it will print out the result of the game according the "flag".

   b) logsInitialization( ):

      This function will initialize and drawing the logs on the map. Notice the length of the logs are fixed that is 15. However, the staring positions for the logs will be randomly set using the rand( ) function.

   c) logs_move(void * t)

      This function will control the logs' and frog's movements according to the parameter "t".

      i. if "t" == 1

         In this case, the function will control the logs' movements. Every time before logs start to move. The function will use usleep( ) to control the refresh rate of the movements. After that, it will use rand( ) function to generate random numbers for each log to make them moves in their direction randomly. Then, the function will check whether the frog will cross the boundary if it is on the logs. If the frog does not cross the boundary, then the function will use pthread_mutex_lock( ) function to lock the mutex and moves the logs according to their random generated moving step. After moving the logs, the function will print the map use printMap( ) function using pthread_mutex_unlock( ) to unlock the mutex.

ii. If “t” == 0

In this case, the function will control the movements of the frog. To make sure if the map modified properly after moving the frog, it will use a char “lastStep” to record the original char in the map at te frog;s position. During the function, it will first lock the mutex using pthread_mutex_lock( ) function and use kbhit( ) to check whether the player has type something to move the frog. If the player types correct instructions, the function will move the frog accordingly. Also, it will check whether the frog will cross the boundary after moving. The function will also check whether the game is win or lose or normal after properly moving. Notice that, the frog will be safe on the riverbank which means it is set not able to cross the boundary on the riverbank. If the game will continue after moving the frog and the position of the frog is changed, the function will use printMap( ) to display the modified map. After finishing corresponding functions, it will use pthread_mutex_unlock( ) to unlock the mutex. For the frog movement, the function will detect the arrow characters. Once the function detects these inputs, it will simply ignore them.

Notice: the mutex will be locked once the function will change the map and unclock it after finish corresponding operations. In this way, the function can make sure that the logs and frog will move properly, that is, the frog won’t move until the logs finishing moving one step and also the logs won’t move until frog finishing moving. Also, the mutex will make sure that the frog will move one step each time.

d) printMap( )

This function will print the chars of the map. It will use printf("\033[H\033[2J") to clear the terminal every time it prints the map.

2. Problems met during the implementation

When implementing the functions to move the logs, I wanted to make sure the frog won't move until all the logs finish moving. Initially, I decided to create 9 threads to move the logs for each log. However, it will be a little complex to make sure that the thread of frog will wait until all the logs moving exactly once. Then I found that it will be a lot easier to achieve the goal by just using one thread to move all the logs. In this case, all the logs will move exactly once during one iteration which means every time the frog move, the logs will just finishing moving for some iterations.

3. Environment

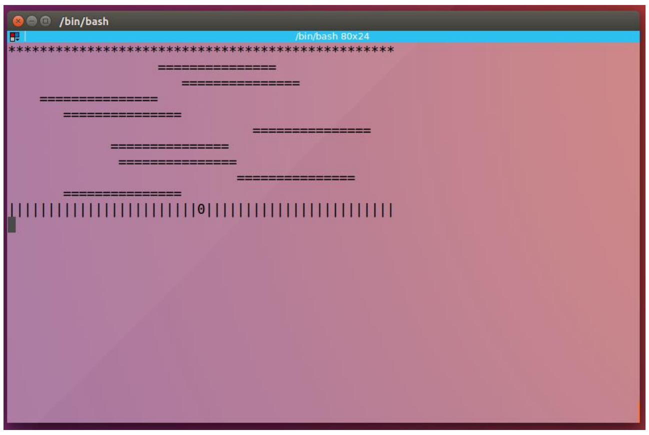
   version of OS: Ubuntu 16.04.2

   version of kernel: 4.10.14

4. Steps to execute the program
   a) First, enter the folder contains the "hw2.cpp" file.
   b) Use "g++ hw2.cpp -pthread" in the terminal to compile the file.
   c) Type "./a.out" to execute the compiled file.

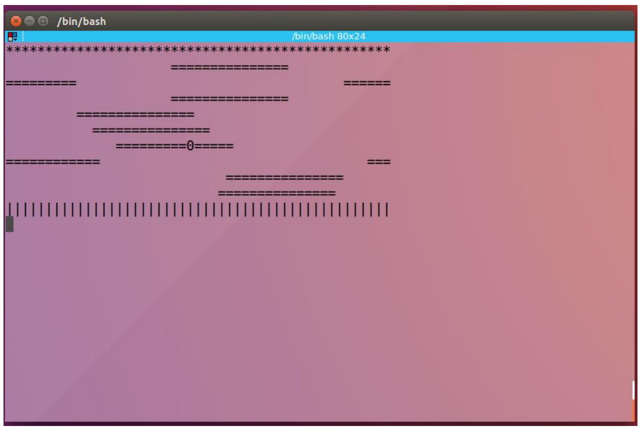
5. Screenshots
   a) During the game:

      i.



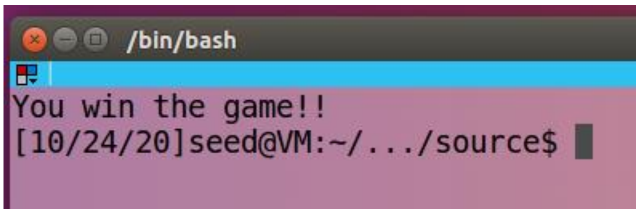
```
*************************************************
                   ===============
                      ===============
    ===============
       ===============
                               ===============
             ===============
              ===============
                             ===============
       ===============
||||||||||||||||||||||||0||||||||||||||||||||||||
```

      ii.



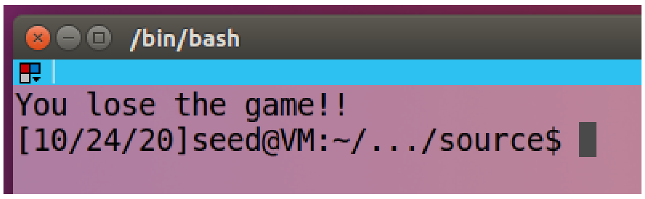
```
/bin/bash
/bin/bash 80x24
*************************************************
                     ===============
=========                                  ======
                     ===============
         ===============
           ===============
              =========0=====
============                                  ===
                            ===============
                           ===============
|||||||||||||||||||||||||||||||||||||||||||||||||
```

b) win the game



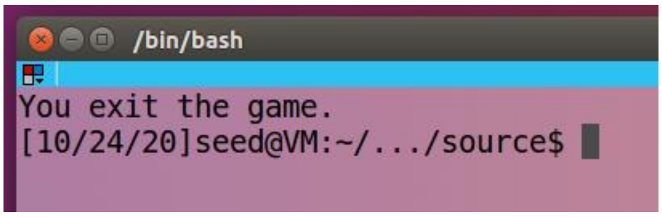
```
/bin/bash
You win the game!!
[10/24/20]seed@VM:~/.../source$
```

c) lose the game



```
/bin/bash
You lose the game!!
[10/24/20]seed@VM:~/.../source$
```

d) quit the game



```
/bin/bash
You exit the game.
[10/24/20]seed@VM:~/.../source$
```

6. What I learnt from the project

First, I learnt how to create threads in the program and control them to do some simple tasks. Also, I learnt how to use “mutex” and “join” to cooperate the threads to implement some simple functions.