EECS201000
Introduction to Programming Laboratory

Homework 3: Arcade Games

Due: July 24, 2017, 8AM

1 GOAL

This assignment aims to get you familiar with pthread programming and the share memory synchronization mechanisms by implementing two arcade games: roller coaster and frogger. In this assignment, you only need to run on the **frontend server**. The **grading is only based on the correctness of your program**, not performance.

2 ROLLER COASTER

2.1 PROBLEM DESCRIPTION

Suppose there are n passengers and one roller coaster car. The passengers repeatedly wait to take rides in the car, which holds C passengers. However, the car can go around the track only when it is full. The car takes the same number of milliseconds (T) to go around the track each time it fills up. After getting a ride, each passenger wanders around the amusement park for **threadID milliseconds (from 1 to n)** before returning to the roller coaster for another ride. Write a program using Pthread to simulate this problem. The program should ask for n, C, T and N, and generate n passenger threads and one roller coaster car thread. The program should exit after roller coaster car going around N times.

- In the passenger threads, you need to simulate as described below.

1. Passenger wanders around the amusement park for **threadID milliseconds.** (use usleep() function to wait).
2. The passenger will return for another ride.
3. Repeat step 1 and 2 until program exit.

- In the roller coaster car thread, you need to simulate as described below.

1. Repeatedly checking if there are *C* passengers in the queue.
2. When there are *C* passengers in the queue, the car will take *T* milliseconds to go around.(use usleep() function too, but wait for *T* milliseconds)
3. Release passengers.

4. Repeat step 1, 2 and 3 *N* times and exit the program.

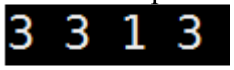
2.2 INPUT

./roller *n C T N out*

- ./roller: your execution file
- *n*: number of passengers (2≤*n*≤10)
- *C*: capacity of car
- *T*: time for car going around the track, representing *T* milliseconds, is integer
- *N*: number of simulation steps
- *out*: name of output file

2.2 OUTPUT

You should print value of n C T N in first line to output file like below:



```
3 3 1 3
```

- In the **passenger** threads,

(a) When a passenger is going to wander around the amusement park, you should print one line to output file like "*Passenger 3 wanders around the park*."

(b) When a passenger returns for another ride, you should print one line to output file like "*Passenger 3 returns for another ride at 3000 millisec*."

- In the **roller coaster car** thread,

(a) When car is going to departure, you should print one line to output file like "*Car departures at 15 millisec. Passenger 1 2 3 are in the car*."

(b) When car arrives, you should print one line to output file like "*Car arrives at 45 millisec. Passenger 1 2 3 get off the car*."

Passenger's **ID number** starts from 1.

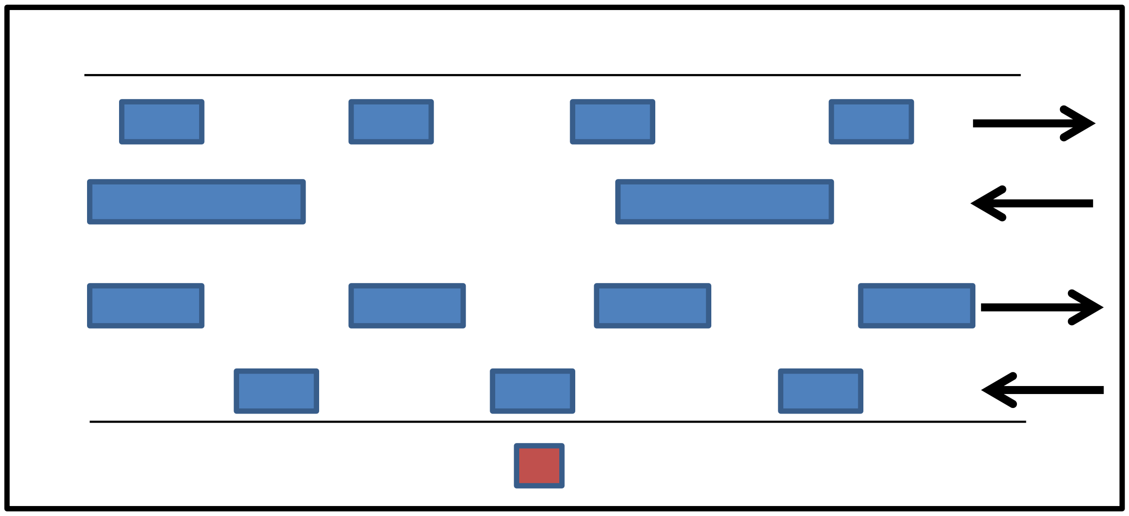
We provide some **sample output** under /home/ipl2017/shared/hw3/

Same input parameter may not generate same output because of threading, so you just need to make sure your output is reasonable (passenger arrival time, car departure time…).

3 FROGGER

3.1 PROBLEM DESCRIPTION

Write a program that plays the game Frogger **on screen**. In this game, you control a frog that is trying to cross a busy **4-lane highway**. Each lane has cars or trucks (indicated by **different sizes of boxes**) zooming by. The vehicles in a given lane all travel at the same speed, but **vehicles in different lanes may travel at different speeds (and even in different directions if you would like)**. The user is in control of the game by **moving the frog up or down using keyboard**. The goal is for the user to get the frog across the highway without it getting squished. If the frog does get squished it should display an "OUCH!" message in the screen, and ask if the user would like to restart the game.



An example of the game can be found here: http://www.retrogames.cz/play_769-Atari2600.php

4 GRADING

1. **Correctness for roller coaster** (45%)
   - TA will verify the correctness using a set of testing inputs.
   - The points are only rewarded if all the tests are passed.
2. **Correctness for frogger** (45%)
   - It is tested in homework demo session.
3. **Bonus for frogger** (20%)
   - Points are rewarded according your additional features in the game. Examples include the follows.

- Allow users to choose from different difficulty levels.
- Better graphic or user interface.
- Add a time restriction requirement.
- Allow frog to more right and left as well.

4. **Demo** (10%)
   - Each student is given 5 minutes to explain your implementation followed by some questions from TA.
   - Not debugging or code modification is allowed during the demo.
   - Points are given according to your understanding and explanation of your code, and your answers of the TA questions.
5. **Late Policy**
   - Receive 80% points after correction within 3 days.
   - Receive 60% points before 8/7.

5 SUBMISSION

- Please upload the following files to **HW_submission/HW3** directory on **apollo31** under your home directory before **7/24(Mon) 8:00AM**: **(The folder will be locked after deadline)**

  i、 **HW3_roller_{student-ID}.c, HW3_frogger_{student-ID}.c**

  ii、 **Makefile**

  We provide a sample makefile under /home/ipl2017/shared/hw3 for your reference, please change ID of HW3_roller_s105062553.c and HW3_frogger_s105062553.c in makefile to your ID.

  Make sure your compile script can execute correctly by and your code has no compile error in the **uploaded folder**.

6 REMINDER

1. You only need to use the **frontend (login) node** in this assignment.
2. **0 will be given to cheater** (even copying code from the Internet), but discussion on code is encouraged.
3. Asking questions through iLMS is welcomed!