Project 3: Interview **CS 300, Hong, 2025 Spring**

Description

An interview booth has two parts, the recruiter sits behind a table in the inner room. He interviews one student at a time, each interview takes a random amount of time. The outer room has *n* chairs for waiting students. When there are no students waiting, the recruiter will focus on his own work behind a big screen. When a student enters the booth and all chairs are taken, the student leaves the booth and will return after studying for a random time period. When a student enters the booth, finds recruiter is talking to a student, but chairs are available, the student sits on one of the empty chairs. If a student enters the booth, finding the recruiter is busy on his work, the student raises the recruiter's attention, so he will talk to the student.

Write a program to coordinate the activities of the recruiter and the students. You will use threads to simulate the interactions between students and the recruiter. Use the synchronization primitives learned from class, such as locks, semaphores, etc., to implement a solution. No other synchronization primitives are allowed.
You will also write a report on this project.

Some Programing Details

The project will use two types of threads, the Students and the Recruiter. Each student will run as a separate thread. The total numbers of students should be input from command line. The recruiter will run as a separate thread. A student thread alternates between waiting for interview or studying for random amount of time. The number of chairs should be input from command line.

To simulate cases such as a student study for a while in student threads, or recruiter talks to a student in the recruiter thread, a sleep function for a random time period should be used. A customized function called int mytime (int left, int right) is provided for this purpose. mytime (int left, int right) produces a random number within a given interval between left and right. It calls rand() for a random number. To have different random number sequences for each execution, one should invoke srand(*z*) to set a proper seed *z* in the main program before any invocation of rand().

After all, the program will read four integers from command line. They are: the number of students, the number of chairs, and the left and right of the interval for the random time period, which will be used in mytime (int left, int right). If input is wrong, prompt should be given for try again with correct format.

Sample code for starting multiple threads is given in BB, file name: PC-inputs-main.c. It is modified from a code example from OSTEP for the Producer and Consumer Problem. In the code, example of thread sleeping is also provided, e.g., thread sleeps for a random time after being created. mytime() is used to get the random time. You can build your project from this sample code.

Additional Requirements:

(A1) Suppose the recruiter allows two opportunities for the same student to talk to him. Then, each student thread terminates after having had the second chance.
(A2) In main program, don't forget join student threads. After all student threads terminate, the main program cancels the recruiter's thread by calling pthread_cancel() and then entire program terminates.
(A3) Please pay attention to:
    (1) The example uses an array for one type of thread, also a loop and sleep after each thread is created. Make sure that when using the random sleep duration, set a short sleep time.
    (2) You need to call the specific function mytime() (see files mytime.c, mytime.h) to obtain a random time to be the input to the sleep function. The makefile is provided.
    (3) A student thread sleeps when arrival finding no chairs are available; the recruiter thread sleeps when finding no students are there.
(A4) In the code, print out necessary info about the activities during the execution of the code.
    (1) Before sleep, print out "Student (or Recruiter) <ThreadID> to sleep X sec;".
    (2) After wake up from sleep, print out "Student (or Recruiter) Id <ThreadID> wake up;".

(3) For either the student or Recruiter thread, before calling mutex locks, or wait on semaphores, printout "Student (or Recruiter) <ThreadID> will call mutex_lock / sem_wait /   <synch variable name> ".

(4) For either the student or recruiter thread, after calling to unlock mutex locks, or post on semaphores, printout "Student (or Recruiter) <ThreadID> call mutex_unlock / sem_post /   <synch variable name> ".

(A5) Feel free to use more functions wherever you see fit. The printouts should be used following the same format given in (A4).

Compiling and Testing

1) Name your code as your P3-sem-*YourFirstLastName*.c, including comment lines at the beginning of the code to show your full name, CWID and the project info.
2) Include lines of #include <pthread.h>, #include <semaphore.h> and #include "mytime.h". *makefile* is provided. If you are compiling multiple files, feel free to compile using provided makefile (as sample). Otherwise, you will use -c to indicate compiling only.
3) Compiling in the makefile uses -Wall -std=c99, which will also be used in grading. Grading uses the cs-intro.ua.edu server. You should try to use the same compiling commands at the cs-intro server to compile your code. Errors lead to points drop.
4) Inputs to the programs: N students, M chairs, the sleep time interval [left, right]
5) Testing program with multiple combinations of numbers of students and chairs, such as: case 1 (1, 1, [left, right]); case 2 (2, 1, [left, right]); case 3 (3, 2, [left, right]); case 4 (3, 5, [left, right]), etc….
6) The running program will stopped by Control-C
7) You can use **Valgrind** to help debug your code and avoid the compiling problem

Some Report Details

1) Write a short report (around 3 pages) to describe the flow of the program (your threads part) and your testcases. Including screenshots of the outputs and a brief explanation of your understanding on the outputs from different testcases. Include at most four cases showing different combination of input numbers, and time intervals.
2) Name your report as *YourFirstLastName-P3.pdf* (.docx is fine). Include your name, CWID, course # in the report.

Submission and Delay policy:

1) Submit only the source code you write. Other source code will be supplied by us at grading time.
2) 10 points are allocated to the submission and compiling.
3) Follow the course policy on delay. Each one-day delay will drop 20% points. You should start your projects early to avoid potential last minute issues that cause late turn-in.

Grading will check:

(1) Submission and Compilation; (2) Correctness of the executions based on the clearness of the outputs from your code; (3) Code inspection; (4) Correctness and clarity of your report.