CS/SE 4348.0U1 Operating Systems

Project 2: Due June 22nd

A university computer science department has a teaching assistant (TA) who helps undergraduate students with their programming assignments during regular office hours. The TA's office is rather small and has room for only one desk with a chair and computer. There are three chairs in the hallway outside the office where students can sit and wait if the TA is currently helping another student. When there are no students who need help during office hours, the TA sits at the desk and takes a nap. If a student arrives during office hours and finds the TA sleeping, the student must awaken the TA to ask for help. If a student arrives and finds the TA currently helping another student, the student sits on one of the chairs in the hallway and waits. If no chairs are available, the student will come back at a later time.

Using POSIX threads, mutex locks, and semaphores, implement a solution that coordinates the activities of the TA and the students. Details for this assignment are provided below.

The Students and the TA

Using Pthreads (Section 4.4.1), begin by creating n students. Each will run as a separate thread. The TA will run as a separate thread as well. Student threads will alternate between programming for a period of time and seeking help from the TA. If the TA is available, they will obtain help. Otherwise, they will either sit in a chair in the hallway or, if no chairs are available, will resume programming and will seek help at a later time. If a student arrives and notices that the TA is sleeping, the student must notify the TA using a semaphore. When the TA finishes helping a student, the TA must check to see if there are students waiting for help in the hallway. If so, the TA must help each of these students in turn. If no students are present, the TA may return to napping.

To simulate students programming in students threads, and the tutor providing help to a student in the tutor thread, the appropriate threads should sleep (by invoking sleep()) for a random of time (up to three seconds).

The total number of students, the number of chairs, and the number of times a student seeks the tutor's help are passed as command line parameters as shown below

```
./tutor #students #chairs #help (e.g. ./tutor 10 4 5)
```

Once a student thread takes the required number of help from the tutor, it should terminate. Once all the student threads are terminated, the tutor thread be terminated, and finally the main program should be terminated.

Print appropriate messages that show the current state of the student/tutor. This will help in both debugging, and validating your program.

Grading Policy

Name your program file as TA.c/cpp and submit it through elearning.

Implementation: 50%. Source code should be structured well with adequate comments clearly showing the different parts and functionalities implemented.

Correctness: 50%.