COMP 521/L—Advanced Operating Systems and Lab
Assignment #4— The Sleeping Teaching Assistant Problem
(A variation of the classic Sleeping Barber Problem)

Objective:

To simulate a teaching assistant helping a random arrival of students (or sleeping during a break) using threads.

Scenario:

- A university computer science department has a teaching assistant (TA) who helps undergraduate students with their programming assignments during regular office hours.
- There are *x* chairs in the hallway outside the office where *y* 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.

Implementation:

- Using POSIX threads, mutex locks, and semaphores, implement a solution that coordinates the activities of the TA and the students.
- Assume *y* students, where each student will run as a separate thread.
- The TA will run as a separate thread.
- Student threads will alternate between programming for a period of time and seeking help from the TA.
- If a student arrives and notices that the TA is sleeping, the student must notify the TA using a semaphore.

Compiling/Running:

- To compile/run your code, you can use one of the following methods:
  - Edit your code using a simple editor (Notepad, Notepad++, etc.) and copy your code into the online compiler: **onlinegdb.com** and click "Run" (if errors, you can use the debugger: "Debug")
  - Edit your code using a simple editor (as above) and compile it using a C compiler installed on your laptop (gcc, Visual Studio C, etc.)—for Mac/Linux users, you can install (if not already) the gcc compiler.
  - Edit your code using emacs/vi/vim, etc. either under native Linux or an image of Linux (ubuntu, Debian, etc.) under VirtualBox/Vmware, and compile using gcc (**gcc filename.c)** and executing using **./a.out**
- Demo your program by running it on your laptop in front of the instructor before or on the due date.

Sample Output (will vary each test run):