**Programming Project 3 – Sleeping Teaching Assistant**

**(Due April 29th)**

In this project, you will practice with mutex locks, semaphores and pthreads. The program will be written in C for the Linux platform.

Read Project 1 – The Sleeping Teaching Assistant Problem on Page 251 of the book. You are allowed to design in groups. The code must be your own work.

**Assignment Rubric (100 points)**

1. In your main header block, list the classmates that you worked with if any. If you choose not to work with anyone, indicate that as well (2 points)

2. Use of header file (ta.h) that contains at least the following variables set as described below (8 points). A header file is available if you wish to use it in Sakai:

/\* the maximum time (in seconds) to sleep. For example, can be used to simulate time helping students, etc. \*/

#define MAX\_SLEEP\_TIME 5

/\* number of maximum waiting students \*/

#define MAX\_WAITING\_STUDENTS 3

/\* number of potential students \*/

#define NUM\_OF\_STUDENTS 5

/\* number of available seats \*/

#define NUM\_OF\_SEATS 3

Thread Management (20 points)

3. Create a pthread for the TA (5 points)

4. Each student (5 students) runs in a separate thread.(5 points)

5. Use of pthread\_join (5 points)

6. When all students are done, cancel the TA thread (5 points)

Program Design/Organization (20 points)

6. Makefile included with all and clean options (5 points)

7. Modular functionality decomposition (at least main.c, ta.c, student.c)(5 points)

8. Proper Indentation for code readability (5 points)

9. Proper Error Handling where necessary (5 points)

Documentation and Output (30 points)

10. Header block with name, course number, project number, date and description of assignment (5 points)

11. Comments describing code throughout (10 points)

12. Output file (15 points) – shows that the code is working properly (with print statements) throughout.

Semaphore and Mutex Use (20 points)

Show proper use of semaphores (one is needed for TA and one is needed for students) (10 points)

Show proper use of mutex lock (needed to protect waiting\_students variable (10 points)

One Possible Plan of Attack

1. Review semaphore/ mutex functionality (chapter 5)
2. Write pseudo code splitting up code into 3 sections, main, student\_loop and ta\_loop.
3. Get all threads (5 student, 1 TA) working properly with random sleep and print statements
4. Add student and ta code – Break these tasks down into manageable, testable sections that make sense to you.