COMP3520 Operating Systems Internals Assignment 1 – Synchronization

General Instructions

This assignment is about synchronization. It consists of two **compulsory** tasks:

1. write a multithreaded program to solve the synchronization problem as described in the section "The Problem: Group Lab Exercise"; and
2. answer the assigned discussion document questions (which are provided in a separate document).

This assignment is an individual assignment. Whilst you are permitted to discuss this assignment with other students, the work that you submit must be your own work. You should not incorporate the work of other students (past or present) into your source code or discussion document.

You will be required to submit your discussion document to *Turnitin* for similarity checking as part of assignment submission. The examiner may use other similarity checking tools in addition to *Turnitin*. Your source code may also be checked.

Submit your source code and report to the appropriate submission inboxes in the COMP3520 Canvas website.

The Problem: Group Lab Exercise

There are *N* students in the class to do lab exercises. The students are divided into *M* groups, each having *K* students except the last group which may have less than *K* students. Each student is assigned to a group with a group *id* **by the teacher**.

There is only one lab room and each time it can only hold one group of students. The students can enter the lab to conduct their lab exercise only when their group *id* is called by the teacher. Each group can only do the lab exercise once.

The exercise will be limited to *T* units of time for each group. (In your program a unit of time is assumed to be one second.) A group may take a minimum of *T*/2 units of time to complete the exercise. After completion, the group will leave the room immediately.

The teacher will call the next group of students to enter the lab after the current group reports the completion of their exercise and the room becomes empty.

To implement synchronization between the various threads, you may use mutexes, and condition variables. **However, you must not use any other type of synchronization mechanisms.**

You program needs to ask the user to enter the following parameters:

- *N*: the total number of students in the class;
- *K*: the number of students in each group; (With *N* and *K*, the number of groups can be determined.)
- *T*: the time limit for each group to do the exercise.

As mentioned previously, each student thread is assigned a group *id* **by the teacher thread** (not by the main default thread when the student threads are created).

To check whether your program functions properly,

each **student thread** must print the following status messages wherever appropriate:

- "Student: I am assigned to group [*id*]."
- "Student in group [*id*]: My group is called by the teacher and I will enter the lab now."
- "Student in group [*id*]: We have completed our exercise and leave the lab now."

the **teacher thread** must print the following status messages wherever appropriate:

- "Teacher: Now I start to assign group id [*id*] to students."
- "Teacher: The lab is now available. Students in group [*id*] can enter the room and start your lab exercise."
- "Teacher: Group [*id*] took [*t*] units of time to complete the exercise."

When all students have completed lab exercises, the main thread must print the following status message and then terminate:

- "Main thread: All students have completed their lab exercises. The simulation will end now."

Additional Requirements

Source Code

Your solution must be implemented in the C language. C++ features are not permitted except those that are part of an official C standard.

Your source code needs to be properly commented and appropriately structured to allow another programmer who has a working knowledge of C to understand and easily maintain your code.

You need to include a well-structured and properly commented *makefile* that allows for the compilation of your source code using the *make* command on the School of Computer Science servers.

Testing and Debugging

You are responsible for testing and debugging your source code.

It is crucial that you ensure that the source code you submit compiles correctly on the School of Computer Science servers and that the resulting binary functions as intended. If you submit source code that cannot be compiled on the School servers or if you fail to make a serious attempt, you may be **DISQUALIFIED** from this assignment.

Discussion Document

You are required to answer all assigned questions in a separate written document. The questions and requirements specific to the discussion document will be provided in a separate document.

Other Matters

Marking criteria for the source code and discussion document will be provided separately.

To maximize your chances of realizing your full potential in COMP3520, **please start work on this assignment promptly.**