CS6868: Concurrent Programming
Spring 2014
Assignment 1: Due 22 March 2014, 11:55 pm

Abil N George (CS13M002)
Dennis Antony Varkey (CS14S008)

Problem 1: Student Professor Synchronization

Problem Statement

You have been hired by the CSE Department to write code to help synchronize a professor and his/her students during office hours. The professor, of course, wants to take a nap if no students are around to ask questions; if there are students who want to ask questions, they must synchronize with each other and with the professor so that (i) only one person is speaking at any one time, (ii) each student question is answered by the professor, and (iii) no student asks another question before the professor is done answering the previous one. You are to write four procedures: AnswerStart(), AnswerDone(), QuestionStart(), and QuestionDone(). The professor loops through the following sequence of actions: AnswerStart(); give answer; AnswerDone(). AnswerStart doesn't return until a question has been asked. Each student loops through the following: QuestionStart(); ask question; QuestionDone(). QuestionStart() does not return until it is the student's turn to ask a question. Since professors consider it rude for a student not to wait for an answer, QuestionDone() should not return until the professor has finished answering the question.

Each student is given a unique id. Any student can ask more than one question: the questions are numbered from 1 for each student. The student id and the question number must be printed clearly for each question that is being asked. Also, to show correctness, the professor should say whose question he/she is answering and the question number of that student. There should be at least 25 students and each one can ask at most 10 questions. The questions could be of different difficulties and therefore the professor may take anywhere from 2 time units to 15 time units to answer all the questions. Always, questions take 1 time unit to ask.

Approach

This problem can be handled by creating a professor thread which waits for the students to ask question. Now to synchronize the students we create ‘n’ students which acquire a lock and try to get a token. This token is used to limit only one student thread to access the critical section.
Hence only one thread is able to access the critical section at a time and the professor answer to only only student at a time. After getting the answer to the question if the student still want to ask more questions he would again come, acquire lock and get a new token. This approach is FCFS based which guarantees total fairness. Hence we can say that no student starves at all.

Implementation

In this problem we first create a student scheduler which creates a random number of student threads at a time. We also create a professor thread which waits for student to start asking questions. Now as a student thread is created it acquires a lock on the token specifying variable and gets its position fixed for execution. It also creates the number of questions it wants to ask by generating a random number between one to ten. Hence by now all the students thread generated come and fix their position of execution. Now all the threads check whether it is its chance to execute. If it satisfies the condition then it is given chance to enter the critical section to ask his question. This is implemented in Question_Start function where the student busy waits for its chance. Now the student in the critical section asks its question and sets the conditional variable to tell the professor to answer the question. So the professor busy waits till the student has asked its question and this is done in Answer_Start function. Here the professor comes in action to answer the question. Once the professor has done answering the question it sends another signal to the student telling his question has been answered. The student then updated the token value so that the next student gets its chance to ask the question. If the student still has question to be answer it goes and fixes its new position of execution. Simultaneously the student scheduler creates more number of student threads on its own parallely. And when there are no more students to ask question the professor goes to sleep.

Sample Output

```
Student  8 is ready to ask question
The professor is ready to answer questions
Student  8 is asking question  1
Student 20 is ready to ask question
Student  8 is done asking question  1
The professor is answering question  1 for student  8
Student 17 is ready to ask question
The professor answered the question  1 for student  8
Student  8 got his answer for question  1

The professor is ready to answer questions
Student 20 is asking question  1
Student 13 is ready to ask question
Student  8 is ready to ask question
Student 20 is done asking question  1
The professor is answering question  1 for student 20
Student  9 is ready to ask question
Student  6 is ready to ask question
Student 15 is ready to ask question
The professor answered the question  1 for student 20
Student 20 got his answer for question  1

The professor is ready to answer questions
Student 17 is asking question  1
Student 21 is ready to ask question
Student 17 is done asking question  1
The professor is answering question  1 for student 17
Student  3 is ready to ask question
Student  5 is ready to ask question
Student 10 is ready to ask question
Student 11 is ready to ask question
. . . . .
. . . . .
. . . . .
The professor is answering question  8 for student 23
Student 12 is ready to ask question
The professor answered the question  8 for student 23
Student 23 got his answer for question  8

The professor is ready to answer questions
Student  4 is asking question  8
Student  4 is done asking question  8
The professor is answering question  8 for student  4
```

The professor answered the question 8 for student 4
Student 4 got his answer for question 8

The professor is ready to answer questions
Student 25 is asking question 8
Student 25 is done asking question 8
The professor is answering question 8 for student 25
The professor answered the question 8 for student 25
The professor is ready to answer questions
Student 25 got his answer for question 8

Student 12 is asking question 8
Student 12 is done asking question 8
The professor is answering question 8 for student 12
The professor answered the question 8 for student 12
Student 12 got his answer for question 8

No more questions from students
The professor goes to sleep