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) No more than a certain number of students can be in the office at the same time because the office has limited capacity. (ii) Only one person is speaking at a time. (iii) Each student question is answered by the professor. (iv) No student asks another question before the professor is done answering the previous one. (v) Once a student finishes asking all his/her questions, he/she must leave the office to make room for other students waiting outside the professor’s office.

You are to provide the following functions:

Professor(). This functions starts a thread that runs a loop calling AnswerStart() and AnswerDone(). See below for the specification of these two functions. AnswerStart() blocks when there are no students around.

Student(int id). This function creates a thread that represents a new student with identifier id that asks the professor one or more questions (the identifier given to your function can be expected to be greater or equal to zero and the first student's id is zero).

First, each student needs to enter the professor’s office by calling EnterOffice(). If the office is already full, the student must wait. After a student enters the office, he/she loops running the code QuestionStart() and QuestionDone() for the number of questions that he/she wants to ask. The number of questions is determined by calculating (student identifier modulo 4 plus 1). That is, each student can ask between 1 and 4 questions, depending on the id. For example, a student with id 2 asks 3 questions, a student with id 11 asks 4 questions and a student with id 4 asks a single question. Once the student has got the answer for all his/her questions, he/she must call LeaveOffice(). As a result, another student waiting on EnterOffice() may be able to proceed.

AnswerStart(). The professor starts to answer a question of a student. Print ...

Professor starts to answer question for student x.

AnswerDone(). The professor is done answering a question of a student. Print ...

Professor is done with answer for student x.

EnterOffice(). It is the student’s turn to enter the professor’s office to ask questions. Print …

Student x enters the office.

LeaveOffice(). The student has no more questions to ask, so he/she leaves the professor’s office. Print …

Student x leaves the office.

QuestionStart(). It is the turn of the student to ask his/her next question. Print ...

Student x asks a question.

Wait to print out the message until it is really that student's turn.

QuestionDone(). The student is satisfied with the answer to his most recent question. Print ...

Student x is satisfied.

Since professor considers it rude for a student not to wait for an answer, QuestionDone() should not print anything until the professor has finished answering the question.

A student can ask only one question each time. i.e., a student should not expect to ask all his/her questions in a contiguous batch. In other words, once a student gets the answer to one of his/her questions, he/she may have to wait for the next turn if another student starts to ask a question before he/she does.