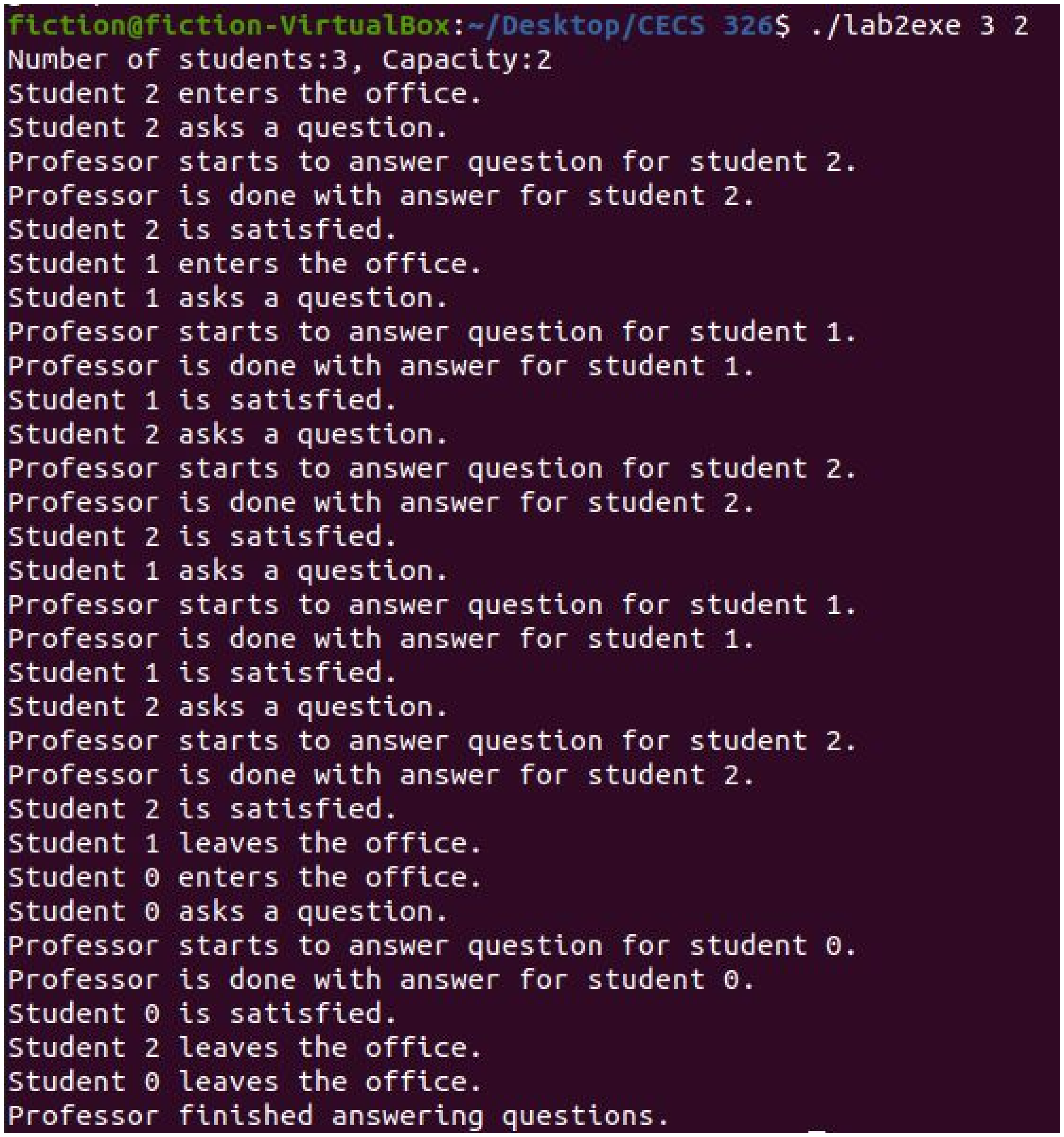
Project 2: Application for Multithreaded Synchronization

Office Hour Program: This program is similar to the producer and consumer method. This program is centered around the professor being the consumer and students being producer. The office will be our buffer and we made sure it does not get too full. If the office is full we tell the next student to wait until the current students are done. Our design revolves around mutex locks, we utilize two global mutex locks and 3 conditional variables. Straight away we know that the Professor is speaking to the student, therefore there must be a lock involved to prevent other students from interfering. By following the rules of this project, we also made a limit for office capacity. Then, we implemented all the given functions; professor(), student(int id), answerstart(). answerdone(), enteroffice(), leaveoffice(), questionstart(), questiondone(). Inside the professor function we will put our first mutex lock and unlock so that the student may start and finish asking their question. For our num of questions we used to formula (student id % 4) + 1. We then used another totalQuestions variable with the same formula. The next lock we use is a lock for studentspeaks inside the Student function; which locks the current student into asking questions then waits for the professor to answer, then unlocks it when the question is finished. Lastly, we use usleep to reset the time for the next student to ask their question to save resources. The wait condition we use in EnterOffice means if the occupancy is equal or greater than the capacity then we tell the office occupancy to wait and lock it. This does not activate in the beginning. In our main function, we made it so that only 2 inputs can be entered when the program starts. All our mutex locks and condition variables are initialized in main. The student threads are created within a for loop then joins together at the end. At the end, the professor thread is joined using pthread_join to represent when all student questions are answered.



```
fiction@fiction-VirtualBox:~/Desktop/CECS 326$ ./lab2exe 3 2
Number of students:3, Capacity:2
Student 2 enters the office.
Student 2 asks a question.
Professor starts to answer question for student 2.
Professor is done with answer for student 2.
Student 2 is satisfied.
Student 1 enters the office.
Student 1 asks a question.
Professor starts to answer question for student 1.
Professor is done with answer for student 1.
Student 1 is satisfied.
Student 2 asks a question.
Professor starts to answer question for student 2.
Professor is done with answer for student 2.
Student 2 is satisfied.
Student 1 asks a question.
Professor starts to answer question for student 1.
Professor is done with answer for student 1.
Student 1 is satisfied.
Student 2 asks a question.
Professor starts to answer question for student 2.
Professor is done with answer for student 2.
Student 2 is satisfied.
Student 1 leaves the office.
Student 0 enters the office.
Student 0 asks a question.
Professor starts to answer question for student 0.
Professor is done with answer for student 0.
Student 0 is satisfied.
Student 2 leaves the office.
Student 0 leaves the office.
Professor finished answering questions.
```

Contributions:
Sam Chen: Wrote the lab report, Helped design code, Debug and fix wrong output of no students being printed, created the makefile

Paul Nguon: Created the readme file, Started designing the code, research, Helped on code to fix the pthreads and condition variables.

ID: Sam Chen, 013502214
ID: Paul Nguon, 015782505
9:02 PM 10/18/20