**Design Document: officehour Simulation**

**Introduction**

This design document aims to provide a detailed explanation of the synchronization mechanisms, algorithm design, and locking structures implemented in the officehour simulation program. The program simulates interactions between students and a Teaching Assistant (TA) during office hours, with a focus on ensuring mutual exclusion, satisfying assignment constraints, and maximizing concurrency.

**Synchronization Mechanisms**

**Mutex and Condition Variables**

Mutex (mutex): The primary synchronization mechanism is a mutex. It ensures that critical sections of the code, where shared data is accessed or modified, are protected from concurrent access.

Condition Variables (ta\_cond, class\_os\_cond, class\_pp\_cond): Condition variables are used for signaling and waiting. Each condition variable corresponds to a specific event (TA availability, OS class student arrival, PP class student arrival).

**Student Counters**

students\_in\_office: Tracks the number of students currently in the TA's office.

class\_os\_inoffice and class\_pp\_inoffice: Counters for the number of students from OS and PP classes in the office, respectively.

students\_since\_break: Counts the number of students served by the TA since the last break.

**Algorithm Overview**

TA Thread (TAthread)

The TA waits for students to arrive in a continuous loop.

Checks if a break is needed based on the number of students served.

If a break is needed, the TA signals all students to leave, takes a break, and resets the counter.

The TA broadcasts availability to students when ready to resume.

Student Threads (class\_os\_student and class\_pp\_student)

Students arrive, wait if the office is full or the TA is on break.

Students ask questions for a specified time.

Students leave the office after asking questions.

Justification of Design Decisions

**Mutex Usage:**

The mutex ensures mutual exclusion, preventing race conditions when accessing shared data.

**Condition Variables:**

Condition variables efficiently manage thread signaling, reducing busy waiting and improving performance.

**Separate Condition Variables for Classes:**

Separate condition variables for OS and PP classes ensure that students from different classes do not interfere with each other, maximizing concurrency.

**Student Counters:**

Student counters help enforce constraints, such as the maximum number of students in the office and the TA break limit.

Satisfaction of Assignment Constraints

**Maximum Number of Students:**

The students\_in\_office counter ensures that the maximum number of students in the office does not exceed MAX\_SEATS.

**TA Break Limit:**

The TA takes a break after serving a specified number of students (TA\_LIMIT), satisfying the TA break constraint.

**Class Separation:**

Separate condition variables for each class ensure that students from different classes do not enter the office simultaneously, meeting class separation requirements.

**Concurrency Maximization:**

The use of condition variables allows students from both classes to independently wait for TA availability, maximizing concurrency.

**Conclusion**

This design document provides a comprehensive explanation of the synchronization mechanisms, algorithm design, and locking structures in the officehour simulation. The chosen approach aims to ensure correctness, satisfy assignment constraints, and maximize concurrency. Regular assertions are used to validate the simulation's correctness during execution.