Program

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define MAX_CHAIRS 3
#define NUM_STUDENTS 5

sem_t ta_available;
sem_t students_waiting;
pthread_mutex_t mutex_seats;
pthread_mutex_t mutex_helped_count;

int waiting_students = 0;
int students_helped = 0;

void *ta_function(void *arg) {
    while (1) {
        sem_wait(&students_waiting);
        pthread_mutex_lock(&mutex_seats);
        waiting_students--;
        pthread_mutex_unlock(&mutex_seats);

        sem_post(&ta_available);

        printf("TA is helping a student\n");
        sleep(rand() % 3 + 1);
        printf("TA finished helping student\n");

        pthread_mutex_lock(&mutex_helped_count);
        students_helped++;
        pthread_mutex_unlock(&mutex_helped_count);


        if (students_helped >= NUM_STUDENTS) {
            printf("TA has helped all students. Exiting...\n");
            pthread_exit(NULL);
        }
    }
}

void *student_function(void *id) {
    int student_id = *(int *)id;
```

```
    sleep(rand() % 5 + 1);
    printf("Student %d arrives at TA office\n", student_id);

    pthread_mutex_lock(&mutex_seats);
    if (waiting_students < MAX_CHAIRS) {
        waiting_students++;
printf("Student %d is waiting (seats available: %d)\n", student_id,
 MAX_CHAIRS - waiting_students);
        pthread_mutex_unlock(&mutex_seats);

        sem_post(&students_waiting);
        sem_wait(&ta_available);

printf("Student %d is getting help from TA\n", student_id);
    } else {
        pthread_mutex_unlock(&mutex_seats);
printf("Student %d found no available chairs and leaves\n", student_id);
    }
    return NULL;
}

int main() {
    pthread_t ta_thread;
    pthread_t students[NUM_STUDENTS];
    int student_id[NUM_STUDENTS];

    sem_init(&ta_available, 0, 0);
    sem_init(&students_waiting, 0, 0);
    pthread_mutex_init(&mutex_seats, NULL);
    pthread_mutex_init(&mutex_helped_count, NULL);

    pthread_create(&ta_thread, NULL, ta_function, NULL);


    for (int i = 0; i < NUM_STUDENTS; i++) {
        student_id[i] = i + 1;
        pthread_create(&students[i], NULL, student_function, &student_id[i]);
    }

    for (int i = 0; i < NUM_STUDENTS; i++) {
        pthread_join(students[i], NULL);
    }


    pthread_join(ta_thread, NULL);
```

```
    pthread_mutex_destroy(&mutex_seats);
    pthread_mutex_destroy(&mutex_helped_count);
    sem_destroy(&ta_available);
    sem_destroy(&students_waiting);

    return 0;
}
```

1 Text File

```
s23a40@Server-2:~/blab$ gcc -o exp17 exp17.c -p
s23a40@Server-2:~/blab$ ./exp17
Student 4 arrives at TA office
Student 4 is waiting (seats available: 2)
TA is helping a student
Student 4 is getting help from TA
Student 2 arrives at TA office
Student 2 is waiting (seats available: 2)
Student 3 arrives at TA office
Student 3 is waiting (seats available: 1)
TA finished helping student
TA is helping a student
Student 2 is getting help from TA
Student 1 arrives at TA office
Student 1 is waiting (seats available: 1)
Student 5 arrives at TA office
Student 5 is waiting (seats available: 0)
TA finished helping student
TA is helping a student
Student 3 is getting help from TA
TA finished helping student
TA is helping a student
Student 1 is getting help from TA
TA finished helping student
TA is helping a student
Student 5 is getting help from TA
TA finished helping student
TA has helped all students. Exiting...
```