CSE344 – System Programming – Midterm Project Report

```c
struct ProducerCustomer {
    sem_t empty;
    sem_t full;
    sem_t m;
};

struct ReadyLeft {
    struct PlateCounts ready;
    struct PlateCounts left;
};

struct MessHall {
    struct Args args;

    struct ReadyLeft kitchen;
    struct ReadyLeft counter;
    size_t counterLeftCount;
    size_t tablesEmpty;
    size_t studentCount;
    uint8_t *tables;

    struct ProducerCustomer semKitchen;
    struct ProducerCustomer semCounter;
    struct ProducerCustomer semTables;
    sem_t chooseBarrier, addSem;
    int fd;
};
```

Student Name: Ozan Şelte

Student Number: 161044061

In this homework to be POSIX compatible with the latest standart I decided to define "_POSIX_C_SOURCE" as "200809", "_FILE_OFFSET" as "64" and "_GNU_SOURCE".

GCC specification:
Thread model: posix
gcc version 9.3.0 (Arch Linux 9.3.0-1)
Used libraries parameters:
-lm
-lrt
-pthread

(struct MessHall *hall) is shared by shared memory and memory map. PlateCount structs has 3 size_t member P, C, D. The plates on the kitchen and the counter are symbolized as *ready PlateCounts*.

./midterm file is the standerd version and ./bonus file is the bonus version. In report I have plotted graphs for only standart version(without -U and -G).

python3 create_plots.py # It recreates the plots in ./figs folder.

Problem #1:

Supplier must know there is any plate that should be placed to the kitchen. This problem automatically is resolved when v8 has arrives.

Solution #1:

Kitchens left member is only modifying by the supplier. So if all three plate types are 0, the loop must be finished.

Problem #2:

sem_wait functions could be return -1 in case of signal interrupts.

Solution #2:

I wrote a function named semWaitRepeat, it waits again if an interrupt comes.

Problem #3:

Cooks and the supplier have to access same memory location and there is a buffer size, they can run without any deadlocks.

Solution #3:

I implemented Producer Consumer Problem solution with three semaphores.

Problem #4:

Cooks have to know how many plates of each types on the kitchen and the counter at the same time.

Solution #4:

Cooks do not choose the plate to carry before got both kitchen and counter mutexes.

Problem #:5

Every cook must know to go in or not in the beginning of the loop but this operations may cause deadlocks.

Solution #5:

In MessHall struct there is coutnerLeftCount variable. This actualy count the left plates in the kitchen side. Every cook enters the critical section checks it and if available, enters the loop. When inside of the loop they are quitting from critical section.

Problem #6:

Students will got the plates only every three types is available on the counter. They can’t take one of it and give them back.

Solution #6:

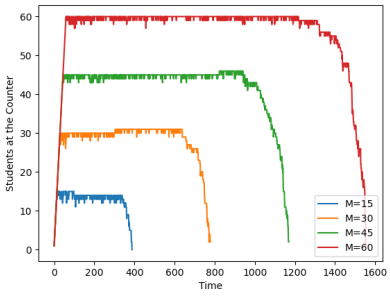
I solved this to create a new pusher process which mixes Producer Consumer and Cigarette Smokers solutions. When a new table to the counter comes, pusher checks the three types and if appropriate it decreases all types 1 than gives running permission to ona of the students.

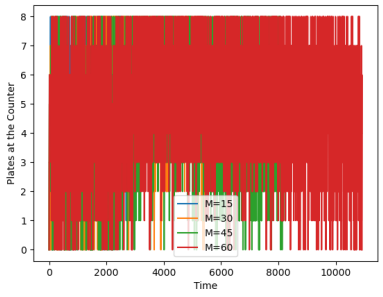
Problem #7:

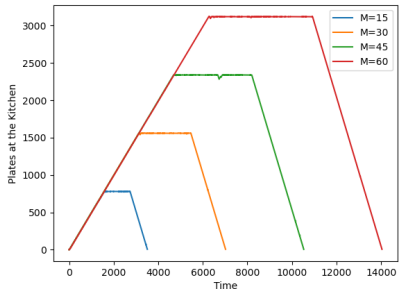
Students must use the tables according to the homework instructions. Thet have to wait an emply table.

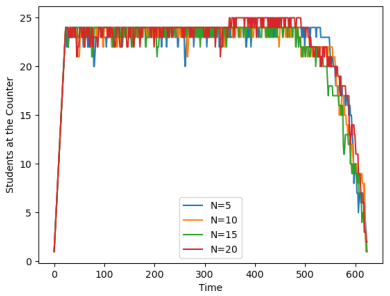
Solution #7:

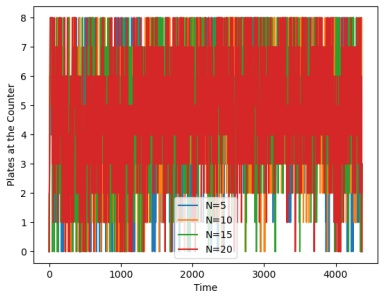
When students both producer and consumer, this problem solved easily.

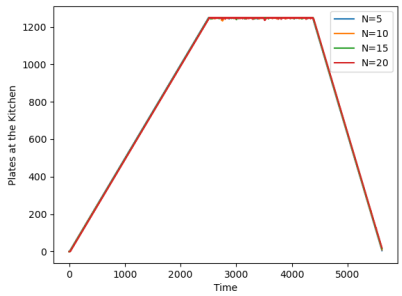


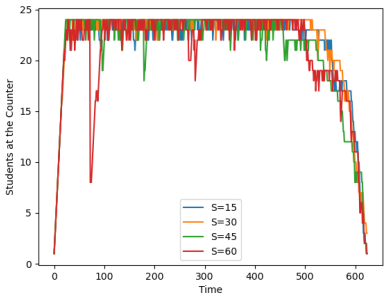


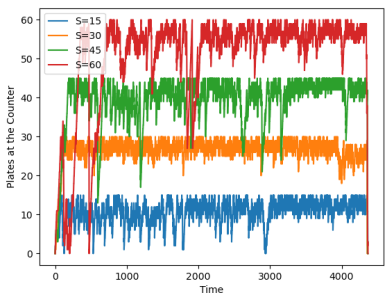


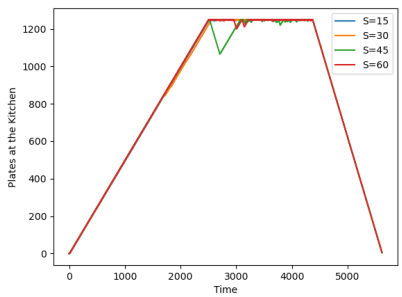


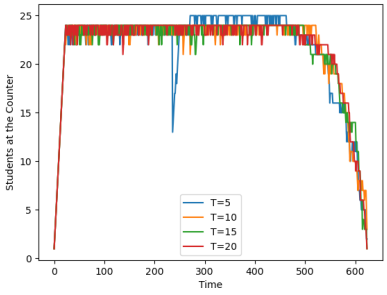


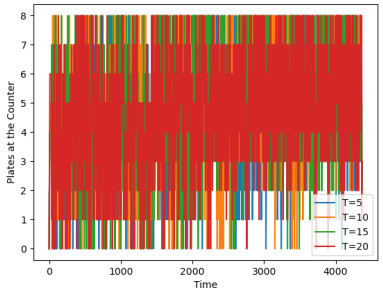


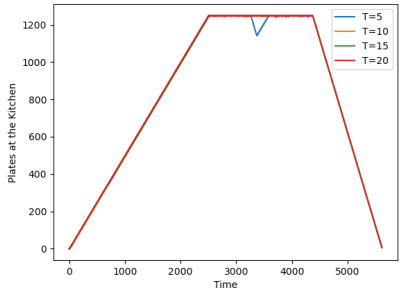


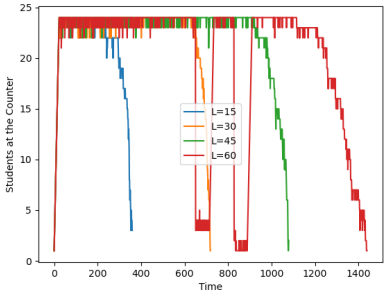


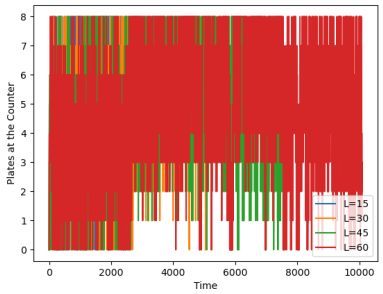


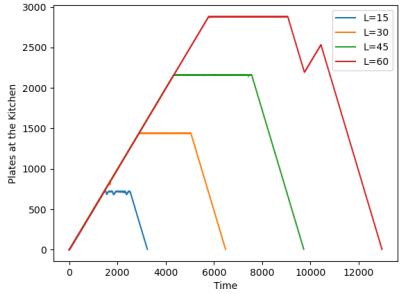

Students at the Counter
Time
T=5
T=10
T=15
T=20


Plates at the Counter
Time
T=5
T=10
T=15
T=20


Plates at the Kitchen
Time
T=5
T=10
T=15
T=20


Students at the Counter
Time
L=15
L=30
L=45
L=60


Plates at the Counter
Time
L=15
L=30
L=45
L=60


Plates at the Kitchen
Time
L=15
L=30
L=45
L=60