Gebze Technical University

Computer Engineering

CSE344 – 2020 Spring

System Programming

MIDTERM PROJECT REPORT

NEVRA GÜRSES

161044071

Erchan APTOULA

Nevra Gürses
161044071

1 INTRODUCTION

1.1 Project Definition

The topic of this project is simulation of mess hall of a university.There are supplier,N cooks and M students.These are 3 different processess.There are also 3 locations that are kitchen,counter and tables.Student number(M),cook number(N),table number(T),counter size(S),how many times a student eat food (L) and file that will read are given as commandline arguments.Kitchen size(K) is calculated as 2LM+1.There are three types of food that are soup,main course and desert.Supplier delivers plates to the kitchen.He/she reads the file that contains exactly 3LM character in an arbitrary order and according to this character, he/she delivers plate to kitchen.If there is no room in the kitchen,the supplier waits for empty room.Until end of the file,the supplier repeats this task.While the supplier process is doing this tasks,it prints 3 types of messages on screen that are about entering the kitchen,after delivery, done delivering.On the other hand,each cook gets a plate from kitchen if there is one,otherwise the cook waits until plates to be delivered by supplier.Then the cook places the plate on the counter if there is any room on counter.If there is no room,the cook waits for empty room.Then the cooks repeats this tasks until all of foods is delivered to the students.While cook process is doing this tasks it prints 4 types of messages on screen that are about waiting for/getting deliveries, going to the counter,after delivery to counter,after finishing placing all plates.And also students wait in front of counter. A student waits until at least one soup, one main course and one desert are available on the counter, in which case she/he takes all 3 of them at once and leaves the counter. After,student searches for empty table.If there is no empty table,the student waits.When he/she finds table,sit down and eat.After eating,the student goes to the counter and repeats this behavior L times.While the student process is doing own tasks,it will prints 5 types of messages on screen that are about arriving at the counter/waiting for food, waiting for/getting a table, sitting to eat, done eating, going again to the counter; times x, where x is increased to x+1, after finishing eating L times. So,in total there are at least N+M+1 processes.

2 METHOD

2.1 Problem Solution Approach

For making tasks of Project,firstly I record all commandline arguments in variables.I make a structure for shared memory,and in that structure I record all semaphores and variables that are using between processess as common.

After, I open an object with a specified name for shared memory with shm_open system call.And I mapped shared memory object with processes virtual address space with mmap system call.Then I initialize all semaphores in shared memory.After,I open input file for reading.Then I create a for loop for supplier,cook and student processess.This loop is 0 to n+m+1.Because there are n+m+1 process.In that loop, I make fork system call and I create child processess.When index is 0,I make tasks of supplier process.I read given file 1 byte 1 and according to read character,I send plate to kitchen if there is empty room in kitchen,if there is no room, supplier waits by using fullKitchen semaphore.I make my operations in critical section between semaphores so I protect operations.I repeat this operations until endof the file.When index is 1 to n,I make tasks of cooks.I take plate if there is plate in kitchen.I take plates in sequence.Firstly I take soup,then main course and after desert.So I send plates to counter in this way and I prevent deadlocks between cooks and students.And I also make operations in critical section between semaphores.When index is n+1 to m, I make tasks of students.Student takes 3 types of plates at the same time if there are avaible on counter.I control whether foods are avaible or not by using values of shared recourses that are semaphores.If all of values is bigger than 0,this means that 3 types of food is avaible on counter,so student takes foods and searches empty table.If there is no empty table,student waits by using fullTable semafore.Once the table is empty,student sits,eats food and then goes to counter to next round.So I make my Project using shared memory,many semaphores in shared memory and processes that are created via fork.

2.2 Synchronization Problems I Encountered and Solutions

- First synchronization problem that I encountered was between supplier and cooks.These problem occured when the supplier and cooks were in the kitchen at the same time.In that case I encountered unwanted situation.For solving this problem,I thought that kitchen was shared resource and operations that were doing in was sharing between processess so there was critical section.So I provided at any moment, there was at most one process (supplier or cook) inside a critical section by using mutex.I write critical section between wait(m) and post(m).So I prevented unwanted situations.I apply this,for all shared resources(counter,table etc.).
- Other sycronization problem between supplier and cook was empty or full situation of kitchen.Supplier could deliver a plate to kitchen if there was any empty room.So I controled whether the kitchen was full or not.And also cooks could take plates if there was a plate on kitchen.So,to control this situations,I used fullKitchen and emptyKitchen semaphores for supplier and cooks.In this way,supplier waited if kitchen was full and also cooks waited if kitchen was empty.
- Sycronization problem between cooks and students occured when student had to take all 3 type of foods at the same time.When cook brought always same foods to counter,counter filled .Since 3 types of foods were not avaible on counter, no student could be able to eat food and so deadlocks occured.For solving this problem,I provided to deliver plates in sequence to counter.So always 3 food could be avaible on counter.
- Other sycronization problem between cooks themselves.There was many cooks.If one of them was working,other cooks was creating deadlock problems because there was many shared resources.So I used a mutex that name was protect1.According to this mutex, two cooks could not work at the same time so deadlocks between cooks was prevented.
- Another syncronization problem between cooks and students was empty or full situation of counter .Cook could deliver a plate to counter if there was any empty room.So I controled whether the counter was full or not.And also students could take plates if there was avaible plate on counter.So,to control this situations,I used fullCounter and emptyCounter semaphores for cooks and students.In this way,cooks waited if counter was full and also students waited if counter was empty.

- Sycronization problem between students and table was full situation of table.A student could sit if there was empty table. So I controled whether the table was full or not.To control this situation,I used fullTable semaphore for students and tables.In this way,students waited if tables were full.

2.2 Functions that I Use in My Project

- **void supplySoup();** In this function the supplier delivers soup to kitchen if kitchen is not full.I increase semaphore named soup and semaphore named foodKitchen in critical section.So I send soup to kitchen.I write critical section between semaphores.So I protect critical section and I control if kitchen is full or not by semaphore named emptyKitchen.
- **void supplyMainCourse();**In this function the supplier delivers main course to kitchen if kitchen is not full.I increase semaphore named mainCourse and semaphore named foodKitchen in critical section.So I send main course to kitchen.I write critical section between semaphores.So I protect critical section and I control if kitchen is full or not by semaphore named emptyKitchen.
- **void supplyDesert();**In this function the supplier delivers desert to kitchen if kitchen is not full.I increase semaphore named desert and semaphore foodKitchen in critical section.So I send desert to kitchen.I write critical section between semaphores.So I protect critical section and I control if kitchen is full or not by semaphore named emptyKitchen.
- **void takeSoup(int cookNum);** In this function cook gets soup from kitchen if kitchen is not empty.I decrease semaphore named soup and semaphore named foodKitchen in critical section.So I take soup from kitchen.I write critical section between semaphores.So I protect critical section and I control if kitchen is empty or not by semaphore named fullKitchen.
- **void takeMainCourse(int cookNum);** In this function cook gets main course from kitchen if kitchen is not empty.I decrease semaphore named mainCourse and semaphore named foodKitchen in critical section.So I take main course from kitchen.I write critical section between semaphores.So I protect critical section and I control if kitchen is empty or not by semaphore named fullKitchen.
- **void takeDesert(int cookNum);** In this function cook gets desert from kitchen if kitchen is not empty.I decrease semaphore named desert and semaphore named foodKitchen in critical section.So I take desert from kitchen.I write critical section between semaphores.So I protect critical section and I control if kitchen is empty or not by semaphore named fullKitchen.
- **void giveSoup(int cookNum);** In this function the cook delivers soup to counter if counter is not full.I increase semaphore named soupCounter and semaphore named foodCounter in critical section.So I send soup to counter.I write critical section between semaphores.So I protect critical section and I control if counter is full or not by semaphore named emptyCounter.
- **void giveMainCourse(int cookNum);** In this function the cook delivers main course to counter if counter is not full.I increase semaphore named mcCounter and semaphore named foodCounter in critical section.So I send main course to counter.I write critical section between semaphores.So I protect critical section and I control if counter is full or not by semaphore named emptyCounter.

- **void giveDesert(int cookNum);** In this function the cook delivers desert to counter if counter is not full.I increase semaphore named desertCounter and semaphore named foodCounter in critical section.So I send desert to counter.I write critical section between semaphores.So I protect critical section and I control if counter is full or not by semaphore named emptyCounter.
- **void soupFromCounter();**In this function student gets soup from counter if counter is not empty.I call this function when all 3 types of foods is avaible on counter,so student takes 3 types of food at the same time.I decrease semaphore named soupCounter and semaphore named foodCounter in critical section.So student takes soup from counter.I write critical section between semaphores.So I protect critical section and I control if counter is empty or not by semaphore named fullCounter.
- **void mainCourseFromCounter();**In this function student gets main course from counter if counter is not empty.I call this function when all 3 types of foods is avaible on counter,so student takes 3 types of food at the same time.I decrease semaphore named mcCounter and semaphore named foodCounter in critical section.So student takes main course from counter.I write critical section between semaphores.So I protect critical section and I control if counter is empty or not by semaphore named fullCounter.
- **void desertFromCounter();**In this function student gets desert from counter if counter is not empty.I call this function when all 3 types of foods is avaible on counter,so student takes 3 types of food at the same time.I decrease semaphore named desertCounter and semaphore named foodCounter in critical section.So student takes desert from counter.I write critical section between semaphores.So I protect critical section and I control if counter is empty or not by semaphore named fullCounter.
- **void sigIntHandler(int sigNo);** This is ctrl_c handler.When ctrl_c entered,in this handler,input file is closed,all semaphores in shared memory are destroyed, Sharing memory object is removed by shm_unlink system call and mappings are deleted by munmap system call.
- **int main(int argc, char *argv[]);**In this function,I create n+m+1 process using fork in for loop.If index is zero I make tasks of supplier by calling supplySoup,supplyMainCourse,supplyDesert functions and I read the file.If index is smaller or equal than n,I make tasks of cooks.I use takeSoup, takeMainCourse,takeDesert giveSoup,giveMainCourse,giveDesert functions in this part.I send all foods in sequence by determining according to current index.And if index is between n and m,I make tasks of students.I use soupFromCounter, mainCourseFromCounter,desertFromCounter functions.
  Student takes all 3 types of foods at the same time if all 3 of them avaible and searches empty table.If there is no empty table,waits by fullTable semaphore.If there is a place,student eats and goes counter for new round.

Nevra Gürses
161044071

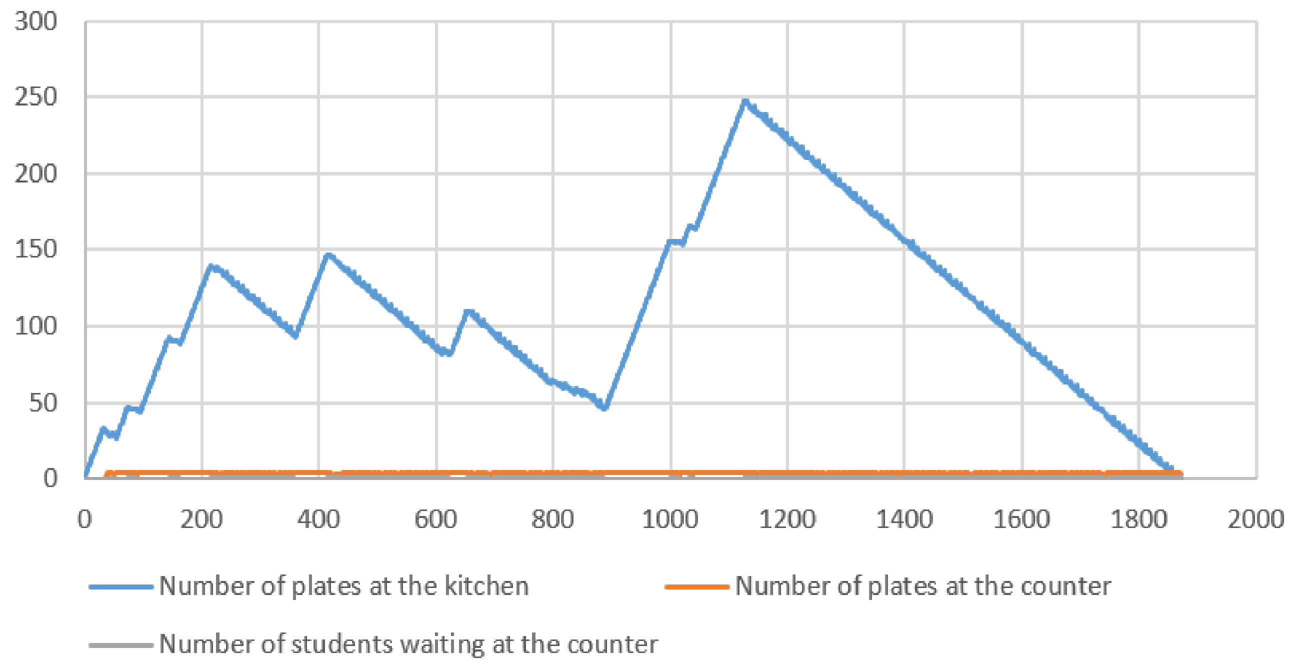
3 RESULT

3.1 PLOTS

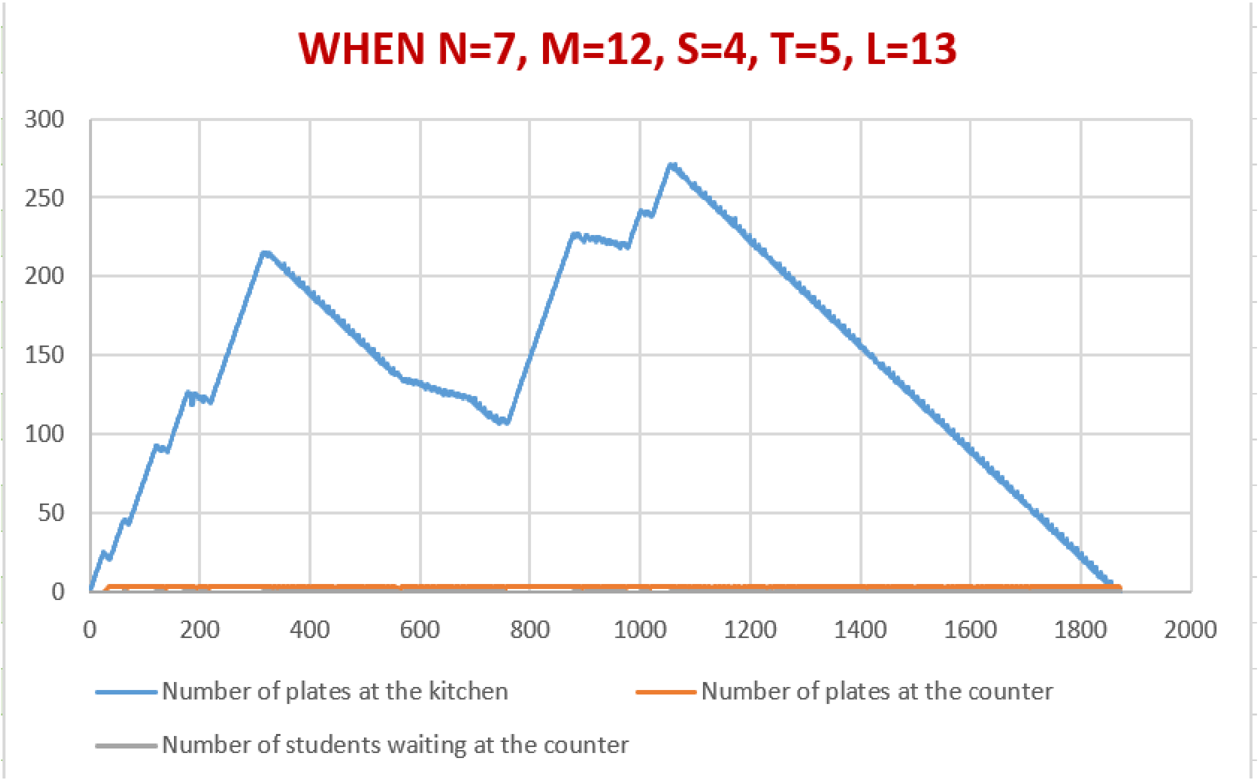
NOTE: In below outputs, Input File is PCDPCDPCD….. :

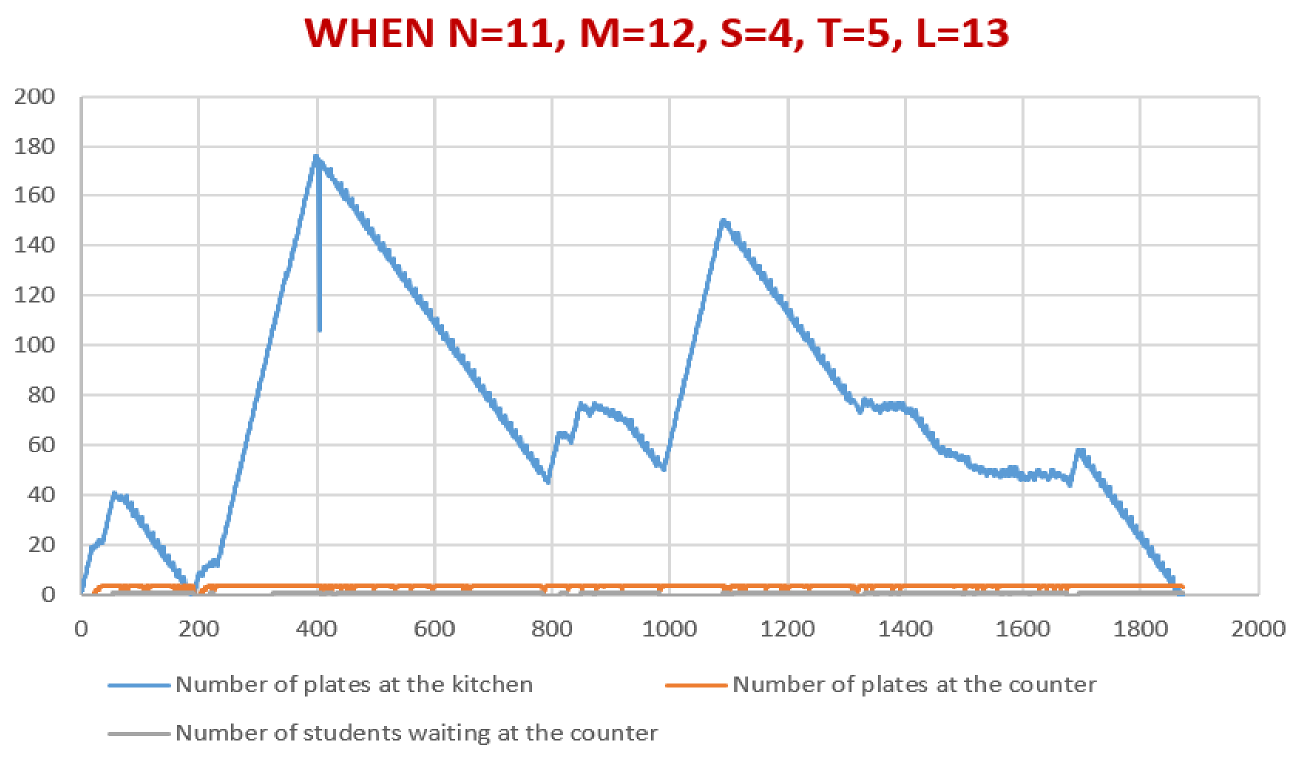
- **If N is changeable, M,S,T,L are fix.**

WHEN N=3, M=12, S=4, T=5, L=13



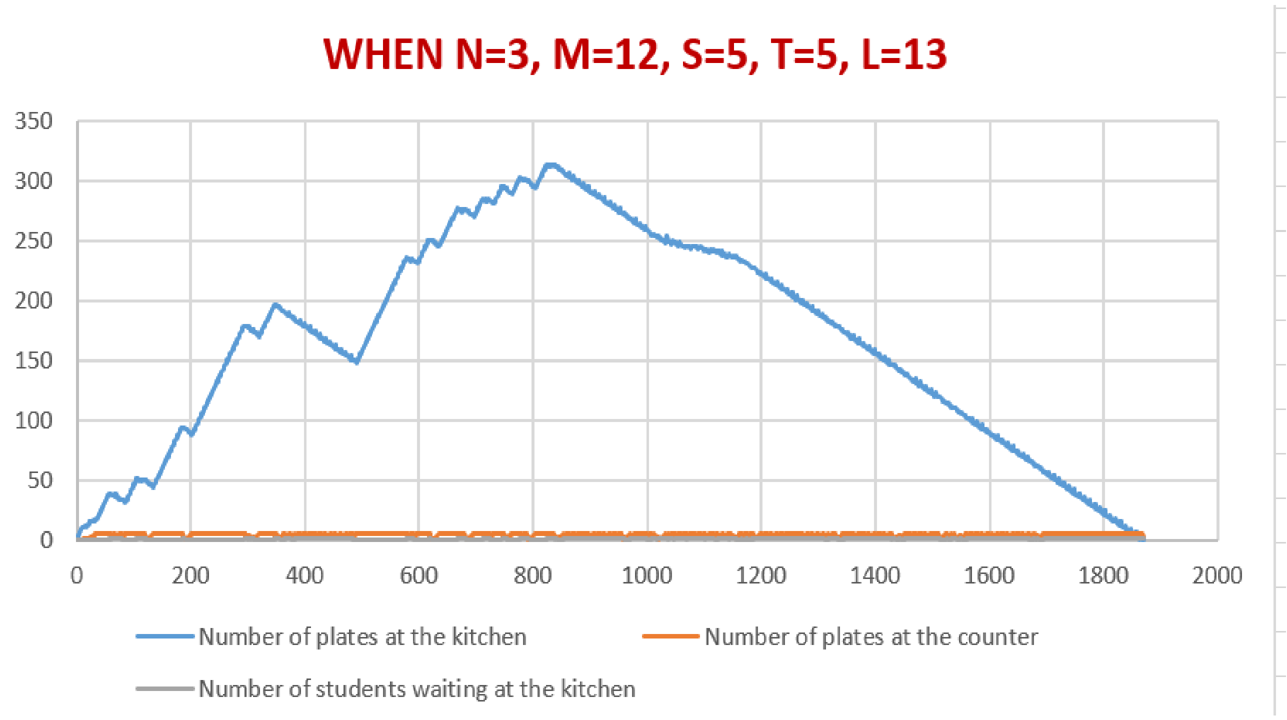
Nevra Gürses
161044071

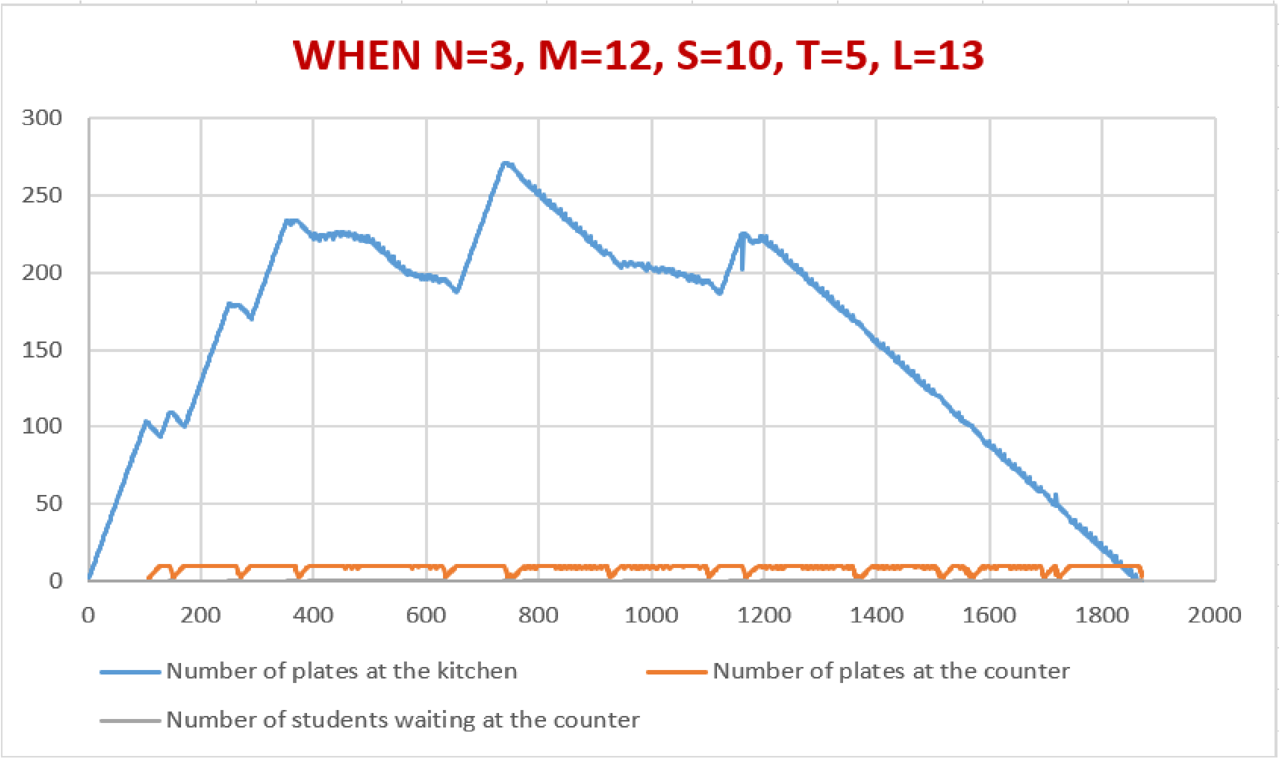


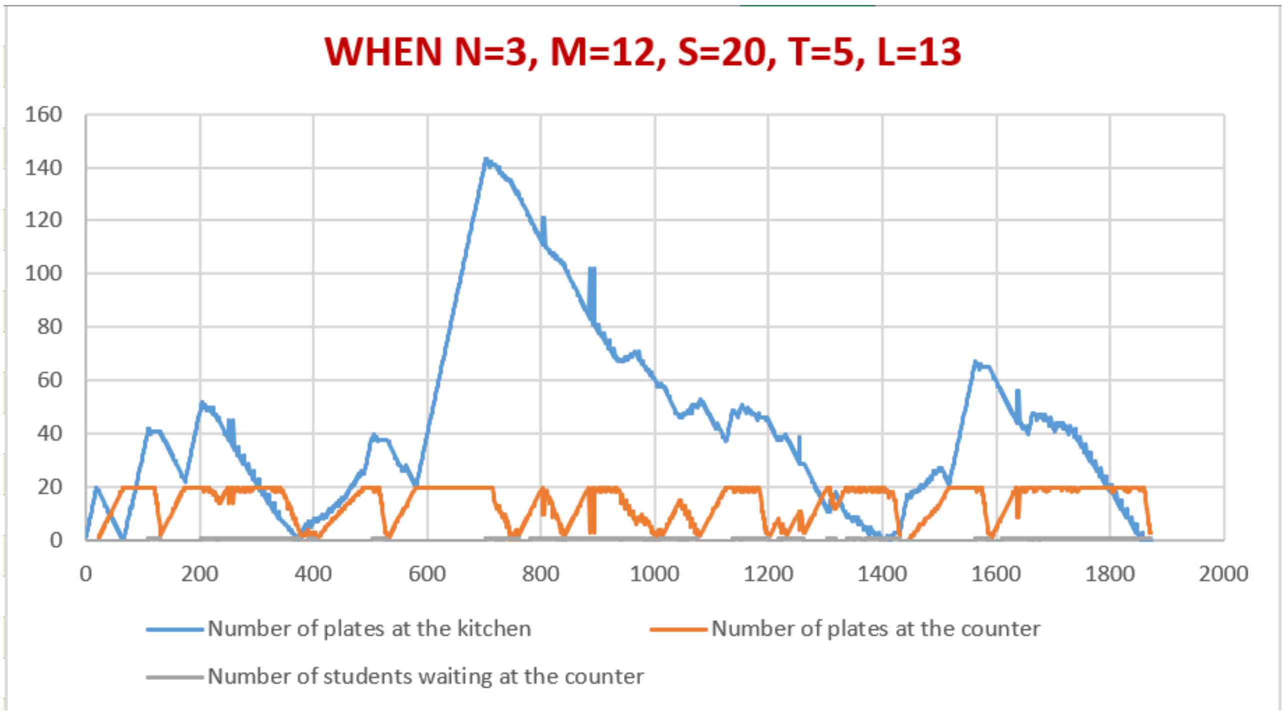


Nevra Gürses
161044071

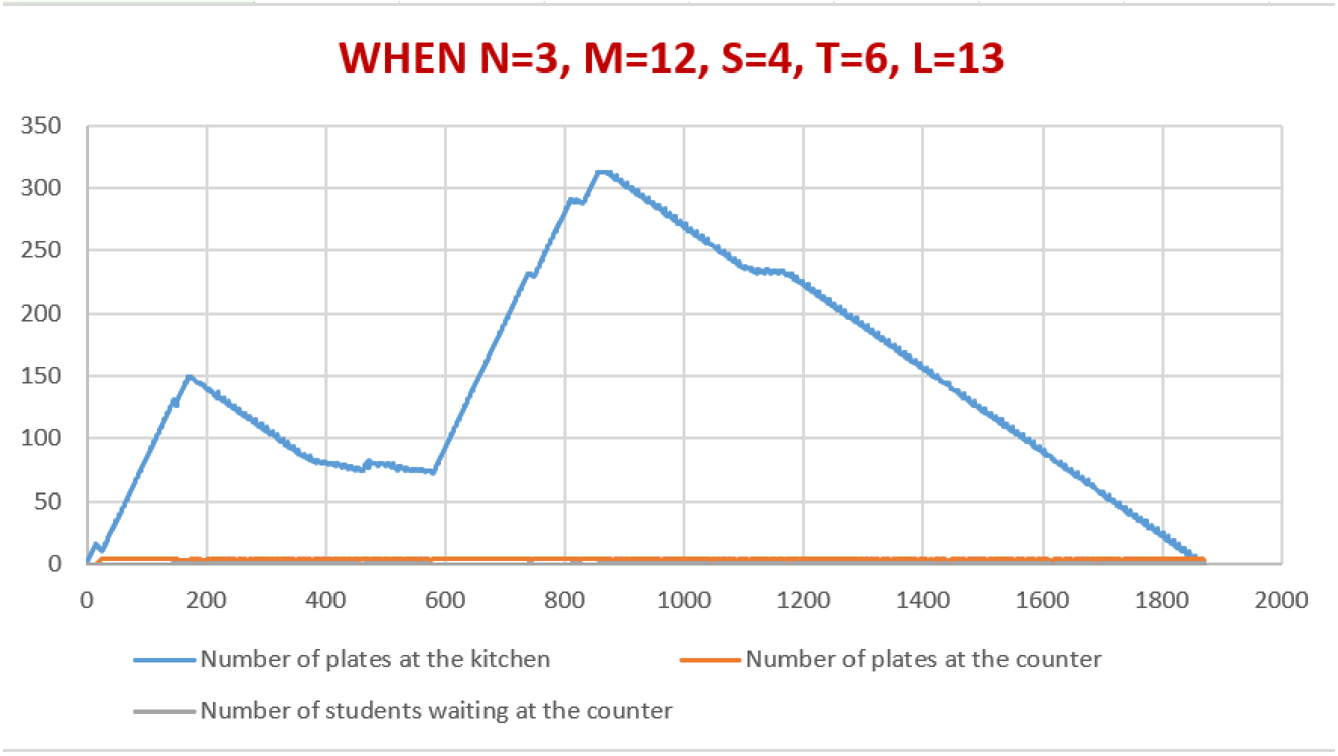
If S is changeable, N,M,T,L are fix.





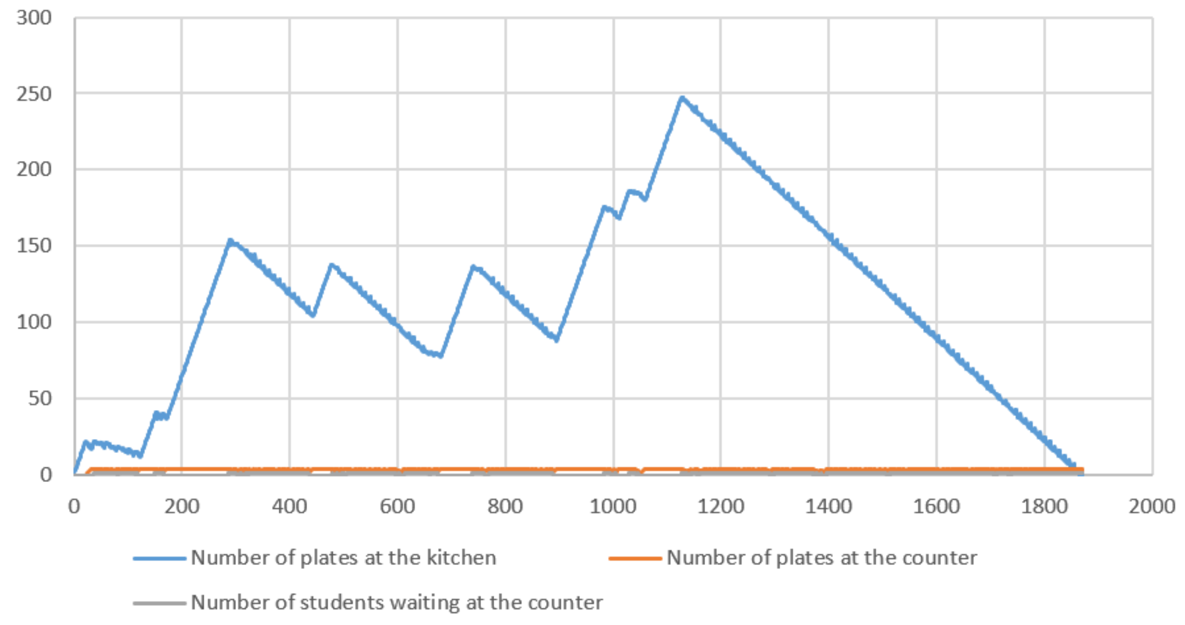


If T is changeable, N,M,S,L are fix.

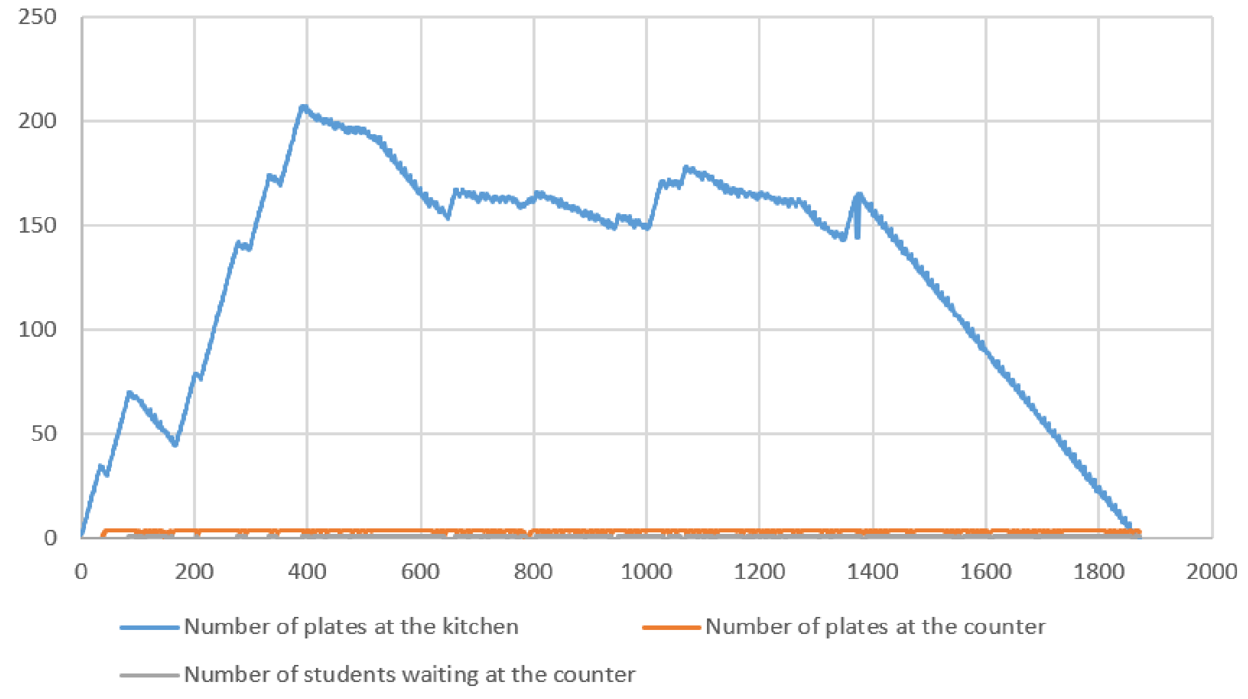


Nevra Gürses
161044071

WHEN N=3, M=12, S=4, T=9, L=13

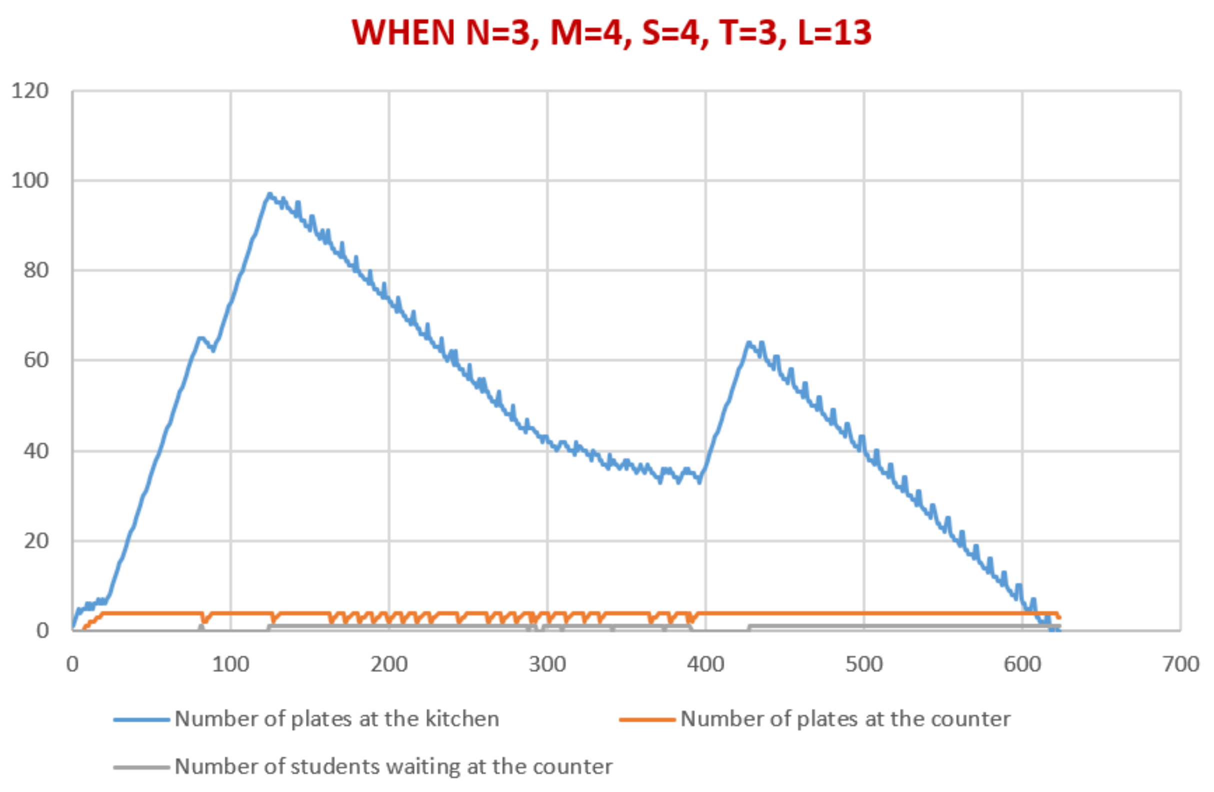


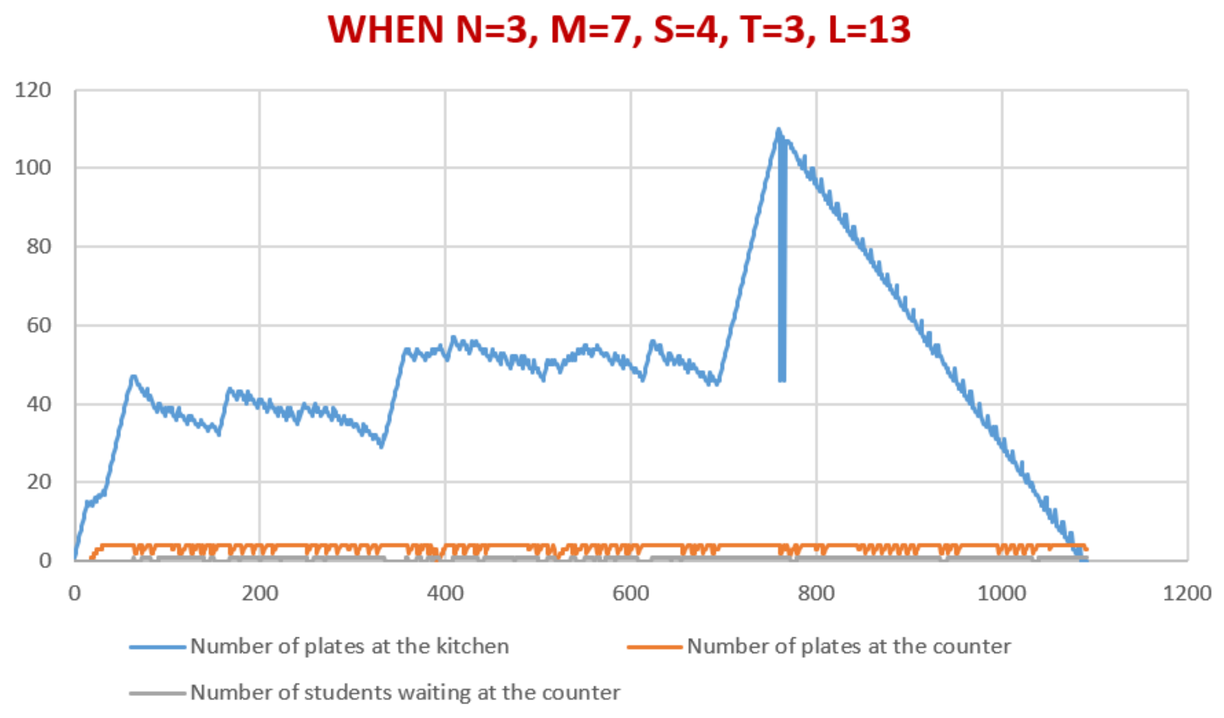
WHEN N=3, M=12, S=4, T=11, L=13



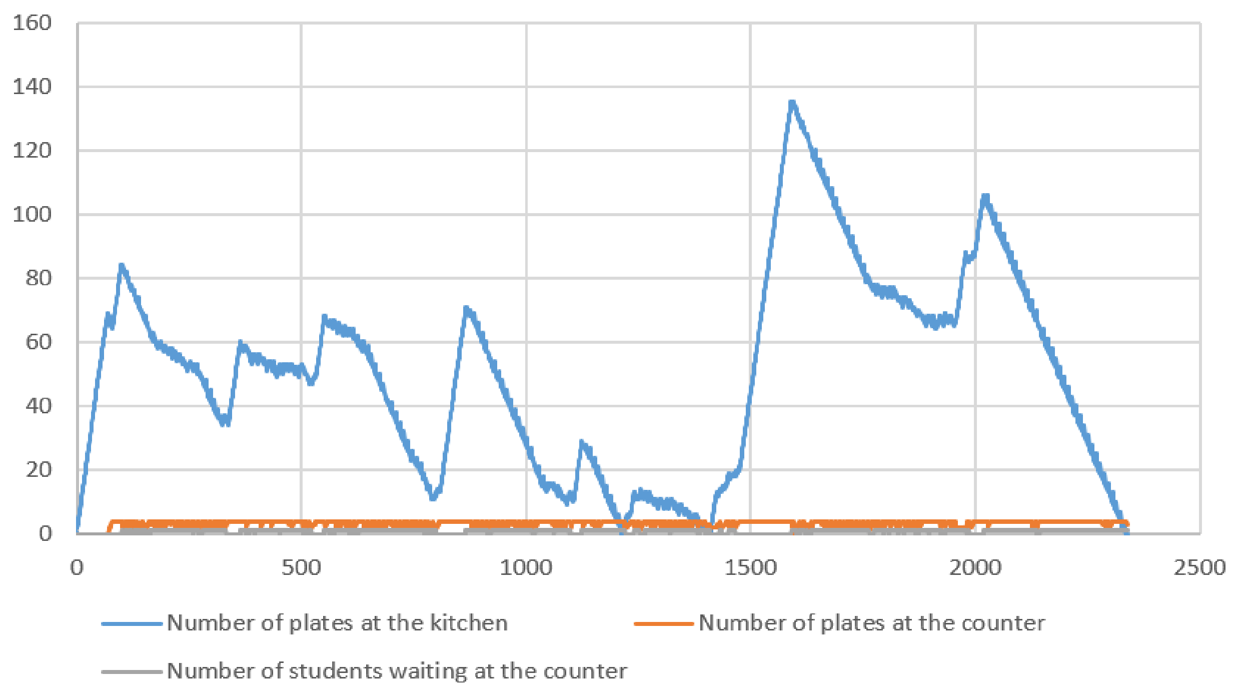
Nevra Gürses
161044071

If M is changeable, N,S,T,L are fix.



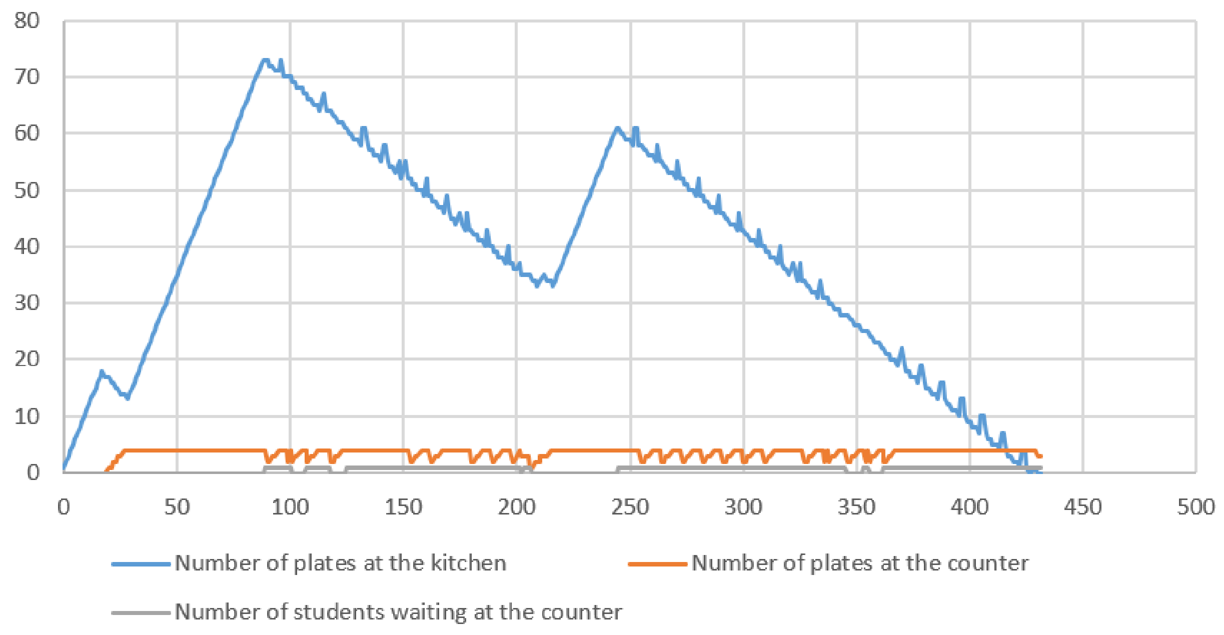


WHEN N=3, M=15, S=4, T=3, L=13



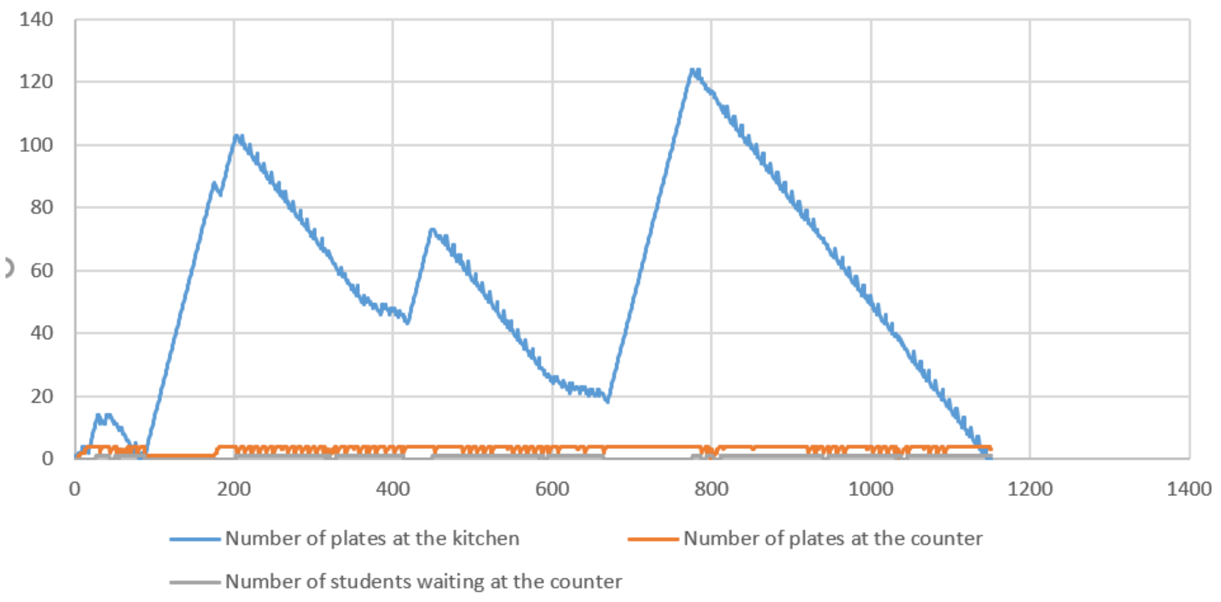
If L is changeable, N,M,S,T are fix.

WHEN N=3, M=12, S=4, T=5, L=3

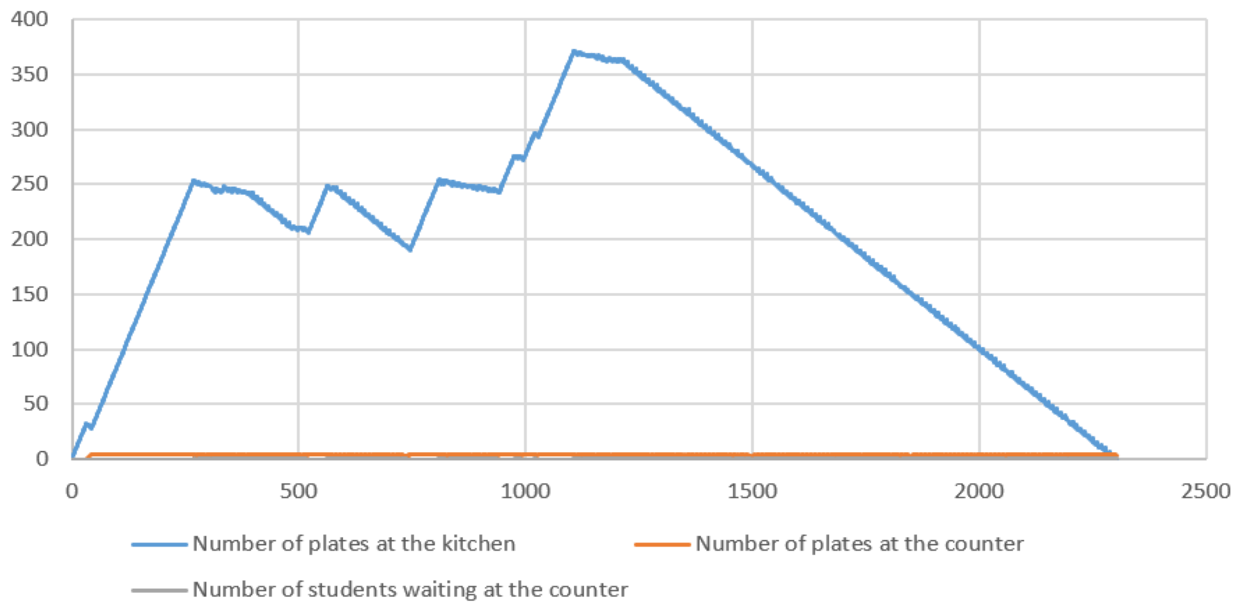


Nevra Gürses
161044071

WHEN N=3, M=12, S=4, T=5, L=13



WHEN N=3, M=12, S=4, T=5, L=16



NOTE FOR COMPILING OF MY PROJECT:

First make command.

Then sudo ./program -N 3 -M 12 -T 5 -S 4 -L 13 -F filename.txt

Sudo command must be writed.