The Korean Restaurant Problem

Ananya Mantravadi
CS19B1004

GOAL

To develop a multithreaded program using semaphores, assuming the number of threads is equal to the number of customers in the restaurant.

ENTRY SECTION

This is the part of the code where customers arrive to join the table. It is accessible to all incoming threads. As all the threads request access to the table/critical section, we allow access to the table only if there wasn't any group formed at its arrival time.

CRITICAL SECTION

This is the part of the code where customers dine. There is a limit on how many customers can access the table at the same time, it is the number of seats available at the table.

EXIT SECTION

This is the part of the code where the customer thread exits from the critical section after finishing dining to allow another thread into it if possible.

IMPLEMENTATION

We initialize a binary semaphores 'mutex' to control customer threads access to the table. Based on whether a group is formed or not, we let all the threads forming a group finish dining before giving access to other threads or let them take a seat at the table simultaneously. The variable bool checks if the group is formed. If so, it will not let new customer threads access the table and dine.

PSEUDOCODE

```
sem_init(&mutex, 0, 1);

Bool group = false;

while (customer threads are left to be given access){

    if(table_size==X){  //group is formed

      group = true;

      continue;

    }

      //wait till start_dine finishes executing and table_size becomes 0

      if(table_size==0){

            group = false;

      }

      if(group==false){

        sem_wait(&mutex);

        table_size++;

        sem_post(&mutex);

        start_dine(curr_thread); //give thread access to table to dine,
will decrement table_size once thread exits

        break;

    }

}
```