**CSE 6431 Programming Assignment**

**Goal:** To master the use of synchronization primitives in either with Java or a C library, and carry out a substantial multi-threaded implementation using these.

**Problem:** Restaurant 6431 is now open. A restaurant requires careful coordination of resources, which are the tables in the restaurant, the cooks available to cook the order, and the machines available to cook the food (we assume that there is no contention on other possible resources, like servers to take orders and serve the orders).

In this project, you will write a simulation for a restaurant. The simulation has four sets of parameters - *the number of eaters (diners) that wish to enter the restaurant, the times they approach the restaurant and what they order, the number of tables in the restaurant (there can only be as many eaters as there are tables at a time; other eaters must wait for a free table before placing their order), and the number of cooks in the kitchen that process orders.*

Eaters in the restaurant place their orders when they get seated. These orders are then handled by available cooks. Each cook handles one order at a time. A cook handles an order by using machines to cook the food items. To simplify the simulation, we make the following assumptions. There are only three types of food served by the restaurant - a Buckeye Burger, Brutus Fries, and Coke. Each person entering the restaurant occupies one table and orders one or more burgers, zero or more orders of fries, and zero or one glass of coke. The cook needs to use the burger machine for 5 minutes to prepare each burger, fries machine for 3 minutes for one order of fries, and the soda machine for 1 minute to fill a glass with coke. The cook can use at most one of these three machines at any given time. There is only one machine of each type in the restaurant (so, for example, over a 5-minute duration, only one burger can be prepared). Once the food (all items at the same time) are brought to a diner's table, they take exactly 30 minutes to finish eating them, and then leave the restaurant, making the table available for another diner (immediately).

**Input:** Diners arrive at the restaurant over a 120 minute duration, which is taken as input by the simulation. *If there are N diners arriving, the input has N+3 lines, specifying, in order: number of diners (N), number of tables, number of cooks, and N other lines each with the following four numbers: a number between 0 and 120 stating when the diner arrived (this number is increasing across lines), the number of burgers ordered (1 or higher), number of order of fries (0 or higher), and whether or not coke was ordered (0 or 1).*

**Your Implementation**: Your simulation must create one thread for each arriving diner, which will then compete for an available table. There will also be one thread for each cook, all of them active for the entire duration when the restaurant is open. Tables and machines for cooking the food are resources whose use must be coordinated among the threads.
Your simulation should output when each diner was seated, which table they were seated in, which cook processed their order, when each of the machines was used for their orders, and

when the food was brought to their table. Finally, you should state the time when the last diner leaves the restaurant. To reduce the length of an experiment, you can use one second to simulate one minute.

**Testing and Grading:** You will be given 5 sample inputs for your simulation. You must submit your code, instructions for executing your code, as well as outputs for the inputs you are given. Note that we will test your code using additional inputs as well, so it is recommended that you consider different situations, prepare additional inputs, and further test your code. Make sure your code can run on the standard configuration of the stdlinux cluster.

You will be graded on correctness (avoiding any incorrect use of resources), efficiency (finishing processing of all diners as early as possible), as well as clarity and readability of the code.

**Submission:** Pack all your source code files, Makefile, and Readme into one gzip file, and submit the gzip file on carmen.