Theory & Programming Assignment 4: The Korean Restaurant Problem

Theory Assignment Submission Date: 24th March 2021, 9:00 pm
Programming Assignment Submission Date: 27th March 2021, 9:00 pm

Problem Statement: This assignment will focus on the famous **Korean Restaurant problem**. Imagine a hypothetical restaurant with a single table, consisting of X seats. If a customer arrives while there is a vacant seat at the table, the customer can take a seat immediately. Whereas, if a customer arrives when all X seats are occupied, this signifies that all X people currently dining form a ***group***. The new customer will have to wait for the entire party to leave before he can sit down.

Please keep in mind that once a customer observes that the table is fully occupied, all subsequent customers have to wait until the table is empty. New customers are not allowed access to the table even if a subset of customers from the group leave the restaurant. An incoming customer can access an empty seat at the table only if the people dining at the table do not form a ***group***.

Example scenarios:

Consider the number of seats at the restaurant table is 5, i.e. X=5.

1) Suppose in the initial configuration, two seats are occupied. Two more customers come in and take seats. Now, suppose one person finishes dining and moves out. Another incoming customer can take a seat since it is available. Note that this can continue as long as no incoming customer finds the table fully occupied (i.e., all the five seats occupied).

2) Similarly, suppose in the initial configuration, two seats are occupied. Three more customers come and occupy the seats. The table is full now.

   As mentioned above, anytime all the five seats are occupied, it is assumed that the guests at the table form a ***group***.

   Suppose 7 new customers arrive. The waiting customers see that the people dining at the table have formed a ***group*** and hence wait. Now, assume one person currently dining at the restaurant leaves. A waiting customer is not granted a seat because the people dining are assumed to be part of a ***group***. The waiting customers have to wait until all the currently dining customers, leave the restaurant and the table becomes empty. After this, a group of five customers from the waiting pool of 7 customers (randomly chosen) will be granted access at the same time. Any new incoming customer will consider these five people as a ***group***.

In this assignment, you are required to develop a multithreaded program in C++ using **semaphores**, keeping in mind the above specifications, for a certain number of customers entering and leaving the restaurant. For simplicity, assume that the number of threads are equal to the number of customers in the restaurant.

There are two parts to this assignment:

1. Theory part: Develop the theoretical solution to this problem and submit it as a theory assignment by the above-mentioned deadline.

2. Programming part: Implement the programming part based on the solution shared.

Theory Assignment Deliverables:

Please submit the solution to the above problem by developing an algorithm using semaphores, keeping in mind the constraints mentioned above. Upload your solution as a pdf named **OS_ThAssign4-<Roll_number>.pdf** within the deadline mentioned in the assignment heading. Kindly do not submit handwritten solutions.

The solution to the theory assignment will be shared after the theory submission deadline which can help you with the programming component.

Programming Assignment Details

Customer Arrival Details:
A set of new customers arrive with an exponential delay, parameter ƛ, equal to 's', where s is uniformly chosen between 1 to r.X. Parameters ƛ, r is provided by the user as an input.

The time each customer eats is exponentially distributed with an average of Γ, after which the customer leaves the dining table. Parameter Γ is also provided as user input.

The program terminates as soon as N customers have dined at the table. Thus, N has to be less than the total number of threads that your system can create.

Sample Input: The input to the program will be a text file **input.txt** containing the total number of customers 'N', the size of the restaurant table 'X' and two parameters ƛ and Γ, each separated by a single white space. The number of threads should be equal to the number of customers in the restaurant.

For example, if N=10, X=5, ƛ=0.1, r=0.3 and Γ=0.2, then the input file **input.txt** should have a single line of the following format:

10 5 0.1 0.3 0.2

System Modelling:

Please note that you have to ensure that the arriving customer threads are successfully created and satisfy the time constraints mentioned earlier. To do this, you can assume that there exists a single master thread that coordinates the thread creation action from the main function of your program. This master thread will make sure that new customer threads arrive at correct intervals and ensure the termination of the algorithm. The customer threads leaving the system however do not need any such monitoring.

Extra Credit: The earlier system model is a centralized approach for the solution to the problem. For an extra 20% credit, please devise a completely decentralized approach for the customer thread creation. Your solution has to work without any master thread as mentioned above.

Sample Output:
Consider a sample output log file of the following form:

```
1st customer access request at 01:00
2nd customer access request at 01:01
1st customer given access at 01:02
..
..
```

The output should demonstrate that mutual exclusion and the conditions of the problem are being satisfied. A few more instructions here:

1. Please don't 'printf' statement as it internally uses locks.
2. Please use cout for printing and use the approach of local buffers as discussed in the class for printing.

Programming Assignment Deliverables: Please submit the source code file as SrcAssgn4-<rollno>.cpp, the output log file, a readme.txt that explains how to execute the program and a report containing an explanation for the working details of the algorithm. Your report should also contain details of the system model that you have used, along with an explanation of the implementation details. The report should also contain the following graphs:

1) Consider the **waiting time** of a customer to be the duration elapsed from the 1st table access request to the time when the customer gets access to the table. Vary the number of threads from 15 to 30, in intervals of 5, and keeping X (the variable) fixed at 4. Plot the average waiting time in microseconds in the Y-axis and #threads on the X-axis.

2) Vary the value of X from 4 to 8, in intervals of 1, and keeping the #threads fixed at 15. Plot the average waiting time in microseconds in the Y-axis and the value of X on the X-axis. Also, find the average waiting time for a customer here.

For both the above graphs, choose the values of λ and Γ according to the system you are working on, such that the average waiting time comes out to be meaningful.

Please zip all the above files and name it as ProgAssgn4-<rollno>.zip. You should then upload it on the google classroom page of this course by the above-mentioned deadline.