Practice Problem Set for Synchronization

As Part of

Operating Systems [CS F372] Course

Semester I, 2020 – 2021

In case of any doubt, please contact: [Reach out to us only after putting in a good amount of effort; we can help you with the logic but NOT with the code!!]

AMOL VIJAYANAND PAI [2017A7PS0038G]
ABOLI VIJAYANAND PAI [2017A7PS0147G]
ANURAG MADNAWAT [2017A7PS1923G]

INDEX

Sec. No.	Section Title	Page No.
1.	**Prerequisites**	**3**
2.	**The Sleeping Barber Problem**	**4**
3.	**The Cigarette Smokers Problem**	**6**
4.	**Reader / Writer Problem**	**8**
5.	**Roller - Coaster Problem**	**9**
6.	**The River Crossing Problem**	**11**

1. PREREQUISITES

- ★ Threads (POSIX threads)
  - ○ pthread_create()
  - ○ pthread_join()
  - ○ pthread_detach()
  - ○ pthread_cancel()
- ★ Semaphores
  - ○ sem_init()
  - ○ sem_wait()
  - ○ sem_timedwait()
  - ○ sem_trywait()
  - ○ sem_post()
  - ○ sem_destroy()

NOTE:

man pages (usually available on all Unix-based systems) are excellent resources themselves; try to use them before you search for something on the Internet

2. THE SLEEPING BARBER PROBLEM

A barbershop consists of a waiting room with N chairs and the barber room containing the barber chairs. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop (without waiting for service). If the barber is busy but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a program to coordinate the barber and the customers. Each customer and barber should be a different thread. Use actual semaphores implementation. Is there a way by which you can evaluate through the execution of your program whether the three conditions of the Critical Section are satisfied?

Extension 1 (Incorporating priority)

Let us say that there are two groups: staffs and students. Number of staffs is N_1 and the number of students is N_2. A waiting student customer has priority over the waiting staff customer (because the students are supposed to be busy !!!).

Therefore, a barber attends to the waiting student customer after finishing the work on the present customer. Modify the code to include this priority condition. Your program should run both with priority and without priority. In priority approach, avoid starvation by enforcing the condition of "each staff can allow a maximum of Y students to overtake".

Extension 2 (Generalization)

Instead of 1 barber, now assume that there are N barbers. Modify the code to include this condition also. (A good code would include the number of barbers as a variable implying the barbershop could hire or fire some of the barbers. Hiring and firing can happen with respect to the number of customers per time).

Extension 3 (Serialization)

The barbershop now has N barbers and a total waiting capacity of M. Out of the capacity of M, there are X sofa seats and the rest (M-X) customers are accommodated in the standing area. A customer will not enter the shop if it is filled to capacity with other customers. Once inside, the customer takes a seat on the sofa or stands if the sofa is filled. When a barber is free, the customer that has been on the sofa the longest is served and, if there are any standing customers, the one that has been in the shop the longest takes a seat on the sofa. Throughout the process, staff can be overtaken a maximum of Y times by a student.

Note 1: The input format is as follows:

Capacity of waiting room (M)
Number of sofas in waiting room (X)
Number of students (N_1)
Number of staff (N_2)
Max overtake allowed by student (Y)

Sample Input:

10 0 10 10 2

Sample Output:

Click for output

3. THE CIGARETTE SMOKERS PROBLEM

PART I:
Consider a system with three smoker processes and one agent process. Suppose a cigarette requires three ingredients: tobacco, paper and match. There are three chain smokers. Each of them has only one ingredient with infinite supply. There is an agent who has an infinite supply of all three ingredients. To make a cigarette, the smoker having tobacco (resp., paper and match) must have the other two ingredients paper and match (resp., tobacco and match, and tobacco and paper). The agent and smokers share a table. The agent randomly generates two ingredients and puts it on the table. The smoker who needs these two ingredients will take them from the table, makes a cigarette, smoke for a while and then exits. Once the ingredients are taken from the table, the agent supplies another two. The smokers wait till ingredients are available on the table. Write a program that simulates this system, with three smokers and the agent being simulated by threads.

PART II:
Consider there are N number of smokers and a single agent. In this case, there can be more than one smoker having the same ingredient at the same time. The priority among the smokers is set on a first come first serve basis. Among the N smokers, you should find out which smokers want the ingredients put by the agent and decide the priority among them. Write a program to synchronize the agent and the N smokers.

PART III:
In this part, the smokers have been classified into two categories, staff and student. The staff has more priority than the student because the student respects the staff and the priorities within the staff and the student is again on a first come first serve basis. To avoid the starvation of the student thread, a student allows only X number of staff to overtake. Modify the program to

i. Synchronize the above processes with only one agent.
ii. Synchronize the above processes with n agents.

PART IV:
Generalize the problem for M ingredients. Each smoker will tell the list of ingredients it needs to create the cigarette.

i. Assume the smoker requests only two ingredients at a time and the agent also puts two ingredients at a time on the table. Incorporate this along with the conditions discussed in the previous parts in your code.

ii. Now assume that the smoker can request any number of ingredients and the agent can also put a random number of ingredients on the table. Solve the problem for this constraint.

Note 1: In the above parts, you can also assume that after a smoker smokes a cigarette, it waits for a while and then joins back in the queue to get the ingredients.

Note 2: The agent is a black box which is only supposed to give the ingredients. Signaling the smoker which has to take the ingredients should be done by some other thread.

Note 3: The index for student/staff given in output is just for checking the position/time at which the smoker came. You can see that the indices of all smokers are in the range [0, N].

Input format:

- The first line gives the number of ingredients (M), the number of smokers (N) and the maximum number of overtakes (X) a student allows.
- The next N lines gives the following information:
  - The type of smoker (Student / Staff) (For the first two parts, assume the type is always Staff).
  - The number of ingredients it requests. (For the first three parts and first sub-part of part IV, the number of ingredients will be 2. For the second sub-part of part IV, this number will be anything in [0, N].)
  - The list of ingredients it wants.

Sample Input:	**Sample Output:**
Part I example	Click for output
Part II example	Click for output
Part III example	Click for output
Part IV-i example	Click for output
Part IV-ii example	Click for output

4. READER / WRITER PROBLEM

Implement a Reader-Writer problem (with ‘N’ readers and ‘M’ writers) with the following constraints:

- Maximum of Y readers can access the critical section at a time.
- If one writer is accessing the critical section, no other writers or readers can access the critical section.
- A writer can allow a maximum of X readers to overtake him while it is waiting for the critical section.
- The problem must take care of precedence (if one writer is executing and ‘N’ readers and ‘M’ writers are waiting for the critical section, the next process to execute must be the first process that came to the critical section).

Implement this with the help of semaphores.

Input format:

- First line contains the values N, M, X, Y – where N specifies the number of readers, M specified the number of writers, X specifies the max number of overtakes for a writer and Y specifies the max number of readers in the critical section.

Sample Input:	**Sample Output:**
#1: 10 10 2 5	Click for output
#2: 10 5 2 3	Click for output
#3: 5 5 0 1	Click for output

5. ROLLER - COASTER PROBLEM

PART I:

Suppose there are N passengers [consisting of N_1 number of kids and N_2 number of adults] [$N=N_1+N_2$] and one roller coaster car. The passengers repeatedly wait to ride in the car, which can hold a maximum of C passengers, where $C << N$. However, the car can go around the track only when it is full. Assume the car ride is for a fixed time [say 5 units]. After finishing a ride, each passenger wanders around the amusement park before returning to the roller coaster for another ride. Due to safety reasons, the car operates only T times and then shuts-off. Suppose threads are used to represent both car and passengers [make sure Name of the car, adult number/ kid number is passed as argument to the thread]. Write a program, using threads and semaphores, which can simulate this system and fulfil the following requirements.

- The car always rides with exactly C passengers [An adult allows M kids to overtake him. Among the adults and among the kids FCFS is strictly maintained].
- No passengers will jump off / on the car while the car is running;
- No passengers will request another ride before they get off the car.
- The passengers are getting down from the car one-by-one [it need not maintain any order].

Note 1: Generate a random number between 1 and 20 and use sleep to simulate wandering time in Amusement park.

PART II:

We can extend this problem by assuming that there are K rollercoaster cars. Modify your code to satisfy the following additional constraints:

- Only one car can be boarding at a time.
- Multiple cars can be on the track concurrently.
- Since cars can’t pass each other, they have to unload in the same order they boarded.
- All the threads from one carload must disembark before any of the threads from subsequent carloads.

Input format: [take it from a file named input.txt]:

Number of kids [N1]
Number of Adults [N2]
Number of Cars [M]
Capacity of the Car [C]
Number of Kids allowed to overtake an adult [M]
Maximum number of trips [T]

Output [store output in output_.txt] [Use proper file operations to store output in output file]

Sample Input:	**Sample Output:**
10 10 1 2 5 3 (Part I example)	Click for output
10 10 2 5 2 5 (Part II example)	Click for output
5 5 3 3 1 3 (Part II example)	Click for output

6. THE RIVER CROSSING PROBLEM

PART I:

There is a rowboat that is used by both Linux hackers and Microsoft employees (serfs) to cross a river. The ferry can hold exactly four people; it won't leave the shore with more or fewer. To guarantee the safety of the passengers, it is not permissible to put one hacker in the boat with three serfs, or to put one serf with three hackers. Any other combination is safe. Write a code to simulate the above situation.

PART II:

Instead of 1 rowboat, assume that there are N rowboats available. The condition in Part I is applicable for each rowboat. Depending on the number of hackers and serfs present, the number of rowboats which should be operational is selected. For example, if there are 5 hackers, 5 serfs, and 5 rowboats present, only 3 rowboats should be operational. Incorporate this condition in your code.

Input format:

The input consists of the number of hackers (N_1), number of employees (N_2) and the number of rowboats (N).

Sample Input:

10 10 1

Sample Output:

Click for output