ASSIGNMENT 3

Submission Date : 03/12/2021
Due Date : 12/12/2021 23:59
TA : Necva BÖLÜCÜ

Accept your *Click here to accept your Assignment 3*.

Instructions: This assignment consists of two parts. The first part involves a series of theoretical questions and the second part involves coding. The goal of this assignment is to make you understand and familiarize with semaphores.

Part I: Theory Questions

1. Ali, Ahmet, and Ayşe plant seeds in a garden. Ali digs the holes. Ahmet then places a seed in each hole. Ayşe then fills the hole up. There are several synchronization constraints:

   - Ahmet cannot plant a seed unless at least one empty hole exists, but Ahmet does not care how far Ali gets ahead of Ahmet.
   - Ayşe cannot fill a hole unless at least one hole exists in which Ahmet has planted a seed, but the hole has not yet been filled. Ayşe does not care how far Ahmet gets ahead of Ayşe.
   - Ayşe does care that Ali does not get more than MAX holes ahead of Ayşe. Thus, if there are MAX unfilled holes, Ali has to wait.
   - There is only one shovel with which both Ali and Ayşe need to dig and fill the holes, respectively. Write the pseudocode for the 3 processes which represent Ali, Ahmet and Ayşe using semaphores as the synchronization mechanism.

   You need to write the pseudocode for the 3 processes which represent Ali, Ahmet and Ayşe using 3 semaphores as the synchronization mechanism.

```
Ali ()
 {while (TRUE) {
     wait(hole);
     wait(shovel);
     signal(shovel);
    signal(empty);
} }
 Ahmet ()
 {while(TRUE){
    wait(empty);
   seed empty hole;
   wait(seeded);
}
semaphore holes = MAX;
semaphore shovel = 1;
semaphore empty, seeded = 0;
Ayşe ()
 {while(TRUE){
    wait(seeded);
    wait(shovel);
     fill hole;
   signal(shovel);
   signal(holes); } }
```

2. We explore the barber shop problem. A barbershop consists of a n waiting chairs and the barber chair. There are several synchronization constraints:
   - If there are no customers, the barber waits.
   - If a customer enters, and all the waiting chairs are occupied, then the customer leaves the shop.
   - If the barber is busy, but waiting chairs are available, then the customer sits in one of the free chairs.

   You need to write the pseudocode for the problem using 3 semaphores as the synchronization mechanism.

```
int sayac=0;
mutex=Semaphore(1);
musterı=Semaphore(0);
berber=Semaphore(0);

void musterı (void){
      wait(mutex);
        if (sayac==n+1) {
          signal(mutex);
        leave(); }

      sayac =sayac+1;
     signal(mutex);

    signal(musterı);
    wait(berber);
    getHairCut();

   wait(mutex);
      sayac = sayac-1;
  signal(mutex); }

void berber (void)
{
  wait(musterı);
  signal(berber);
 cutHair();
}
```

Part II: Dining Philosophers

The Dining Philosopher Problem states that K philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. Philosophers are either thinking or eating. Whenever a philosopher wishes to eat, s/he first needs to find two chopsticks. S/he may eat if he can pick up the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both. If the hungry philosopher does not have two chopsticks (i.e. one or two of her neighbors already picked up the chopstick) s/he will have to wait until both chopsticks are available. When a philosopher finishes eating, s/he puts down both chopsticks to their original places, and resumes thinking. There is an infinite amount of food on their plate, so they only need to worry about the chopsticks.

- Philosophers are either thinking or eating. They do not talk to each other.
- Philosophers can only fetch chopsticks placed between them and their neighbors.
- Philosophers cannot take their neighbors' chopsticks away while they are eating.

Submit You are required to submit all of your code along with a report in PDF format. The codes you submit should be well commented. The report should contain answers of the theoretical questions.

The ZIP file will be submitted via Github Classroom. *Click here to accept your Assignment 3*

1 Grading Policy

1. Code (60): Implementation of Dining Philosophers: 60
2. Report (40): Theory part: q1:20, q2:20

Important Notes

- Do not miss the submission deadline.
- Compile your code before submitting your work to make sure it compiles without any problems.
- Save all your work until the assignment is graded.
- All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.
- You may assume that the input files will be given as:

  - Assignment3.java or Assignment3.c
  - Report.pdf