**OS-2 project**

**Documentation**

**Project Title:** Multiple sleeping barber problem **Group:**

**Discussion time:** 12:20 PM **Instructor:** Mohamed Kamal

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Name (Arabic)** | **Bounce** | **Minus** | **Total Grade** | **Comment** |
| **202000364** | **زينب علي موسى** |  |  |  |  |
| **202000702** | **ماريو أيمن كرم دوس** |  |  |  |  |
| **202000149** | **امنه مصطفى سعد محمود** |  |  |  |  |
| **202000742** | **محمد حامد عبد الحميد عبد النبي** |  |  |  |  |
| **202000810** | **محمد فريد يوسف سيد البحيري** |  |  |  |  |
| **202000601** | **عمر عاطف وجدى عطية** |  |  |  |  |
| **202001001** | **نورالدين على محمد الحنونى** |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Critical |  | | | | | | | | Grade | | | Team Grade | Comment |
| Documentation | Solution pseudocode | | | | | | | |  | 1 | |  |  |
| Examples of Deadlock | | | | | | | |  | 1 | |  |  |
| How did solve deadlock | | | | | | | |  | 1 | |  |  |
| Examples of starvation | | | | | | | |  | 1 | |  |  |
| How did solve starvation | | | | | | | |  | 1 | |  |  |
|  | Explanation for real world application and how did apply the problem | | | | | | | |  | 1 | |  |  |
| GitHub | Upload project files | | | | | | | |  | 2 | |  |  |
| Submitted before discussion time (shared GitHub project link with TA and Dr) | | | | | | | |  | 1 | |  |  |
| Only one contribution | | | | | | | | -1 | | |  |  |
| Implementation | Run correctly (correct output) | | | | | | | |  | 5 | |  |  |
| Run but with incorrect output | | | | | | | | -3 | | |  |  |
| Not run at all (error and exceptions) | | | | | | | | -8 | | |  |  |
| Free from Deadlock | | | | | | | |  | 3 | |  |  |
| Free from deadlock in some cases and not free in other cases | | | | | | | | -2 | | |  |  |
| Free from Starvation | | | | | | | |  | 2 | |  |  |
| Free from Starvation in some cases and not free in other cases | | | | | | | | -1 | | |  |  |
| Apply problem to real world application | | | | | | | |  | 6 | |  |  |
| Total |  | Total grade for Team | | | | |  | |  | 25 |  |  |  |
|  | Total Team Grade(after adjustment) | | | | | |  |  | 25 |  |  |  |
| Bounce | Multithreading GUI Based Java Swing | | | | | | | | +5 | | |  |  |
| Multithreading GUI Based Java | | | | | | | |  |  |
| Swing( | | adjustment | | ) | | | |
| Multithreading GUI Based JavaFX | | | | | | | | +10 | | |  |  |
| Multithreading GUI Based | | | | | | | |  |  |
| JavaFX ( | | | adjustment | | ) | | |
| Bounce Graphic and animation | | | | | | | | +5 | | |  |  |
| Total with  Bounce |  | Total Team Grade | | |  | | | |  | | |  |  |
|  | Total Team Grad (after adjustment) | | | | | |  |  | | |  |  |

**Multiple sleeping barber problem**

**Problem description:**

A barber shop which has multiple barbers working in it, when a barber is not working, he sleeps till a customer arrive, when a customer arrive a barber wakes up, cuts his hair then check whether there are more waiting customers, if there is none he goes back to sleep. The other barbers stay asleep as well if there are no more customers for them to work.

When a customer arrives, he checks if there’s an empty chair for him, if there’s none he leaves else he awakes a sleeping barber to cut his hair.

**An example of deadlock:**

When barbers wait for their customers and at the same time the customers wait for the barbers to call out their names.

**Solution:**

The barbers should check the customer’s waiting room before sleep

**An example of starvation:**

When customers don’t follow an order and customers get a haircut before others even if they wait for a long time, or when a barber keep doing all the jobs leaving others sleeping.

**Solution:**

Giving higher priority to the customer or barber who waited more by enabling fairness in semaphore or lock

**Real-world Example:**

A program to check whether the solution of a problem is correct or not by having a tester thread that checks whether the answer is the same as the correct answer or not, and a submission thread that loads the answer. The submission waits in a queue for a tester to finish working, and testers wait for a submission to arrive to test it like how the barber wait customer and a customer wait for barber.

**Solution Pseudocode:**  
# We use a Semaphores to check whether a Barber or a customer is ready

class BarberShop(chairs, barbers)

#Initialize locks and Semaphores, Fairness = true to avoid starvation

Semaphore waiting\_customers ← Semaphore(0, true)

Semaphore barbers\_ready ← Semaphore(barbers, true)

customer\_lock ← Lock(true)

number\_of\_customer ← 0

Function add\_customer()

lock customer\_lock

if number\_of\_customer = chairs

print(shopFull)

unlock customer\_lock

Return

Else

increment number\_of\_customer

unlock customer\_lock

increment/release waiting\_customers

#Check whether a barber is ready by acquiring Semaphore

aquire barbers\_ready

#Check whether there's a waiting customer by aquiring Semaphore (useless)

aquire waiting\_customers

customer.getHairCut()

barber.Cuthair()

Wait for barber and customer to finish

release barbers\_ready