Kartik Gupta (170101030)
CS431 - Assignment 1

Concurrent Programming

Question 1

Role of Concurrency & Synchronization

Concurrency

There are multiple Robotic Arms in the system that work together to pick up the socks from the heap and pass it to the Matching Machine, Along with that Matching Machine and Shelf Manager are working simultaneously too.

Synchronization

The heap of socks is available to all the robotic arms and there can be instances where various robots try to pick up the same sock so those cases have to be handled and have to be synchronized properly. Matching of the socks present in the matching machine's buffer and addition of a new sock in the matching machine's buffer is done synchronously

How did you handle it?

Concurrency is achieved by multithreading where for each robotic arm, a thread is created which tries to pick up sock from the heap and pass it to the matching machine.

I have used **method synchronization** on a Sock object so that only one Robotic Arm can pick up a sock at a time.

Question 2

Why is concurrency important here?

All the teachers (TA1, TA2 and CC) are independent of each other and they can simultaneously update the marks of the students. In absence of concurrency, others have to wait while one is updating the marks of a student which can be done independently. So concurrency is required for allowing them to concurrently update the marks of different students simultaneously to speed up the complete process.

What are the shared resources?

The input file that contains the information of all the students is the shared resource. The marks can be changed by different people so it needs to be handled with care.

What may happen if synchronization is not taken care of ?

If synchronization is not taken care of, it might lead to inconsistent results as multiple threads could be trying to update the marks at the same time.

For example:
Assume there's a student X whose initial marks were 50
Query 1 - TA1 tries to increase marks of student X by 10
Query 2 - TA2 tries to decrease marks of student X by 10
If both of queries are executed simultaneously, Query 1 will load the student's marks as 50 then update it to 60
Then Query 2 will also load the student's marks as 50 and will update it to 40.
The final result will be 40, which isn't intended. There should be no net change in the marks of the student X.

Such problems will be solved by proper synchronization.

How did you handle it?

Concurrency is achieved by multithreading where a single thread is created for each teacher (CC, TA1, TA2) and these threads are executing concurrently to update the marks.

I have used **reentrantlock** for synchronization. A thread has to acquire a reentrantlock corresponding to the record of the student it wants to modify. This ensures that there is only one thread modifying a particular student's record.