**BÀI TẬP TRÊN LỚP MÔN HỌC: HỆ PHÂN TÁN**

**CHƯƠNG 6: Đồng bộ hoá**

HỌ TÊN SV: Nguyễn Đức Thiên MÃ LỚP: 114175

MSSV: 20168806 MÃ HỌC PHẦN:IT4610Q

***Câu hỏi thực hành:***

**1. Deploy the synchronization for threads of a multithreaded process**

Question 1: The output value is random value from 1000 to 3000 because 3 workers run the exploit function without wait others workers finish their work. Hence, all worker should get the same value of resource at the same time, then increase the resource independently. For example, at time t, worker1 and worker2 call the exploit() function, then the getRsc() function return value for both worker is 100 and then both worker call setRsc() function with same value 101.

A close up of a logo

Description automatically generated

Question 2: The output value of resource is always 3000, because we using synchronized(rExp) that deny multithread access resource at the same time. Each thread must wait until others worker finish their loop with resource.

A close up of text on a black background

Description automatically generated

Question 3: With lock access to resource, worker have the same output as synchronized workers in question2. The lock variable try to lock access to resource in ten seconds, waiting the function exploit to complete and then unlock the resource. So that neither worker2 nor worker3 could access resource while work1 was working.

**2. Parallel Programming with Critical Sections**

Question 4:

A close up of a logo

Description automatically generated

Question 5: Increase NUM\_TRANS to 1000 and use 10 threads, there will be a very clearly difference between balance and INIT\_BALANCE + credits - debits. The difference between two of this balance because there were many threads using the variable balance at the same time without lock access or using synchronization.

A screenshot of a cell phone

Description automatically generated

Question 6: There is a little different between Shared and Expect cause by lock mechanism. There could be several thread waiting lock variable. And when one thread release the lock, others thread should run same time. And because the code using **postfix** (shared++), the value of share will be copy and then increase. Several thread will copy the same value and increase at the same time.

A picture containing monitor, black, screen, holding

Description automatically generated

A close up of text on a black background

Description automatically generated

Question 7: With mutex lock, there are no wrong answer becouse balance was locked each time we calculate balance.

A close up of text on a black background

Description automatically generated

Question 8: Using NUM\_TRANS = 1000 and 10 threads, without\_lock average excuted time is around 0.003 second. Mutex\_lock average excuted time is around 0.005 second, much longer. But the mutex\_lock have an acceptable accuracy.

A close up of text on a black background

Description automatically generated

A screenshot of text

Description automatically generated

Question 9: Both fun\_1 and fun\_2 using lock\_a and lock\_b. It's lead to problem deadlock which happened when fun\_1 release the lock\_a to increase a, but fun\_2 immediately lock it.

A close up of text on a black background

Description automatically generated