HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**OPERATING SYSTEM – LAB 06**

\_\_\_\_\_\_\_

*Student name: LE HUY HOANG*

*Student ID: 1652212*

*Date: October 17th , 2018*

Lecturer: PhD. Pham Hoang Anh

1. Assume Alice is the wife, Bob is Alice’s husband.

There are 3 possible cases in this situation:

1. Alice finishes deposit transaction right before Bob starts withdraw transaction.

In this situation, no conflict occurs.

2. Alice starts deposit transaction right after Bob has finished withdraw transaction.

There is no conflict also.

3. Alice and Bob start each other transaction concurrently.

The account balance depends on which transaction comes first. More details, assume Alice and Bob do their transaction on ATM machine and computer. ATM machine will do as follow:  
 1/ Retrieve account balance  
 2/ Do the transaction

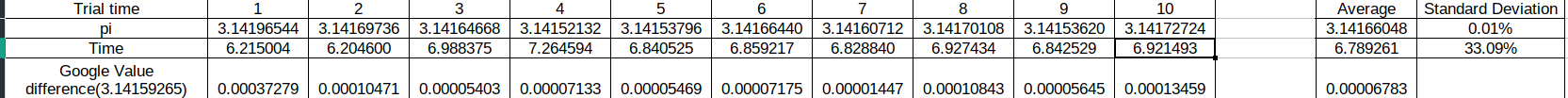
3/ Update the account balance to bank server.

Internet Banking service does the same.

If Alice’s and Bob’s machines retrieve account balance at the same time, then at stage 3, bank server will be misleading in determining appropriate number to update. The final result will depends on the order of incoming update requests from ATM or machine

The solution for this problem is to design a mechanism that mutually exclude incoming requests that could potentially affect the account balance. For example, if it happens, bank server have to randomly choose which requests will be served first. The other must be in waiting queue. Plus, that mechanism must be non-preempty. The reason is bank server does not know whether user does another transaction after finishing one or not. Therefore, other incoming transactions must wait until the serving request finishes its shift.

However, bank server should reject waiting requests since the serving time of the current request is unknown beforehand.

2.

With mutex lock, time is slightly longer in comparison to that of the previous week solution. Here I list the time for each running (100 millions point)