# Lab Task Multithreading Mutex and Synchronization

## I. Four Concurrent and Synchronizing Threads

Write a program that generates 4 threads: two withdrawal threads that continue to withdraw $10 from a bank account and two depositing threads that continue to deposit $11 to the same account. Assume there is no limit on the maximum balance of the account but the withdrawal threads should block if the account does not have sufficient fund (i.e. less than $10) to withdraw. You should allow maximum currency for all threads, i.e., you do not unnecessarily synchronize threads and allow CPU switching after each depositing or each withdrawal.

## II. The Functions for Each Thread

The following code is part of function withdraw(void \*tid) executed by the withdrawal thread (you need to complete it by adding more code on mutex and synchronization if necessary):

*for (int i=0; i<10; i++){*

*int readbalance = balance;*

*printf(“At time %d, the balance for withdrawal thread %d is %d”, i, tid, balance);*

*readbalance -= 10;*

*sleep(1);*

*balance = readbalance;*

*printf(“At time %d, the balance after withdrawal thread %d is %d”, i, tid, balance);*

*sleep(1);*

*}*

The following is part of function deposit(void \*tid) executed by the depositing thread (you need to complete it by adding more code on mutex and synchronization if necessary):

*for (int i=0; i<10; i++)*

*{*

*int readbalance = balance;*

*printf(“At time %d, the balance before depositing thread %d is %d”, i, tid, balance);*

*readbalance += 11;*

*sleep(10);*

*balance = readbalance;*

*printf(“At time %d, the balance after depositing thread %d is %d”, i, tid, balance);*

*sleep(10);*

*}*

The mutex and synchronization variables should be cleaned up after each thread finishes its job.