Assignment 3

EEL4930/5934 Advanced System Programming

Due: Friday, February 22nd by midnight

In this assignment, you are going to simulate electronic fund transfer (EFT) between bank accounts. We will assume that there is just one bank and several accounts. Your program will take an input file in the form

AccountNo1 <space> initialBalance1
AccountNo2 <space> initialBalance2
..
AccountNoN <space> initialBalanceN
Transfer <space> accountNoFrom1 <space> accountNoTo1 <space> Amount1
Transfer <space> accountNoFrom1 <space> accountNoTo2 <space> Amount2
…
Transfer <space> accountNoFrom1 <space> accountNoTo1 <space> Amount1

which first lists the accounts in the system along with the initial balances and then lists the transfers between accounts. You can assume that all transfers refer to existing accounts and all initial balances and the transfer amounts are nonnegative integers.

Your program should take one more parameter to denote the number of worker threads that will run in parallel. Note that the main thread will initialize the accounts, read the input file and assign work (EFTs) to worker threads in a round-robin fashion until all transfers are processed. It is possible that an account may be overdrawn and gets a negative value while processing. Your program should output on the standard output the amount in each account (in the order specified in the input file) once all transfers are computed.

Usage of your program:

$ transfProg inputFile numWorkers

So, as an example, assuming the input file is as below

1 1000
2 50
3 400
4 150
Transfer 1 2 200
Transfer 1 4 50
Transfer 2 3 100

your program should produce the following output regardless of the number of worker threads specified:

1 750
2 150
3 500
4 200

To get full credit, your solution should maximize concurrency, be free of race conditions and deadlocks, and produce the correct output. You can use mutexes and condition variables and/or semaphores for synchronizing your threads. Please submit your files (source, README, and a Makefile) on CANVAS by the due date.