CSE 325 **Spring 2020**

Computer Project #6

Assignment Overview

For this assignment, you are to design and implement a C/C++ program that uses multi-threading to simulate a simple banking system based on the Producer-Consumer paradigm.

It is worth 40 points (4% of course grade) and must be completed no later than 11:59 PM on Thursday, 2/27.

Assignment Deliverables

The deliverables for this assignment are the following files:

`proj06.makefile` – the makefile which produces **`proj06`**
`proj06.student.c` – the source code file for your solution

Be sure to use the specified file names and to submit your files for grading via the CSE Handin system before the project deadline.

Assignment Specifications

1. The program will create P producer threads and one consumer thread which share access to a bounded buffer of size B. The number of producers and the size of the bounded buffer will be available to the program as command-line arguments. Valid executions of the program might appear as:

```
proj06 -p 3 -b 10
proj06 -b 5 -p 4
proj06 -b 20
```

The number of producers P will not exceed 10 and will default to 1. The bounded buffer will be circular and will consist of B records, where B will not exceed 20 and will default to 5.

2. When the program begins execution, it will read zero or more lines from the input file named "accounts.old" to initialize the set of customer accounts. Each line of that file will contain an account number (integer number) and account balance (real number). When the program halts execution, it will write the updated set of accounts to the output file named "accounts.new".

3. Each producer thread will process the transactions which are contained in the input file named "transN", where N corresponds to the thread number (and thus is between 0 and 9). Each input file will contain zero or more lines, where each line represents a transaction.

Each line of an input file will contain an account number (integer number), transaction type ("deposit" or "withdraw"), and transaction amount (real number). For example, the input file named "trans0" (which will be processed by thread #0) might contain the following lines:

```
189348 deposit 1500.00
519783 withdraw 40.00
```

Each producer thread will repeatedly read one transaction from the appropriate input file and then insert one record representing that transaction into the bounded buffer. After all of the transactions in the appropriate input file have been processed, that producer thread will insert a special record into the bounded buffer to indicate that it is finished processing transactions and will then halt.

4. The consumer thread will repeatedly extract one transaction record from the bounded buffer, than process it. After all of the producer threads have halted and the bounded buffer is empty, the consumer thread will halt.

If the transaction represents a deposit into an account, the consumer thread will add the transaction amount to the account balance.

If the transaction represents a withdrawal from an account, the consumer thread will subtract the transaction amount from the account balance.

The consumer thread will log the results of each transaction by sending one line to the output file named "accounts.log". Each line will be no more than 80 characters in length and will contain:

a) thread number where the transaction originated
b) account number
c) transaction type
d) current account balance
e) transaction amount
f) updated account balance

Those log entries will be appropriately formatted: items will be aligned in columns, and monetary values will be displayed as dollars and cents (for example, $50.00).

5. The program will perform appropriate error handling.

Assignment Notes

1. As stated above, your source code file will be named “proj06.student.c”.

2. You must use “g++” to translate your source code file in the CSE Linux environment.

3. You must use the POSIX threads library for this assignment. Information about system calls and library functions which might be useful for this project may be viewed using the "man" utility. For example:

```
man 3 pthread_create          man 3 sem_init
man 3 pthread_join            man 3 sem_wait
man 3 pthread_exit            man 3 sem_post
```

4. You may assume that the lines in the input file named "accounts.old" (if it exists) are formatted correctly and contain valid account numbers and account balances.

5. You may assume that the lines in the input file named "transN" (if it exists) are formatted correctly and contain valid account numbers and transaction types. You may assume that the transaction amounts are positive values.

6. You will model the bounded buffer using an array and you will model one item in the bounded buffer using a record (C/C++ "struct").

7. You may model the set of customer accounts using any data structure (including data structures from the STL).

8. Each critical section will be kept as small as possible and no I/O operations will be performed inside a critical section.