Lab 1 - "Non-cooperative" multithreading - Documentation
Bank Accounts

Overview
This program simulates a banking system where a number of operations (transfers) are performed between accounts. The accounts' balances are updated concurrently by multiple threads, making synchronization necessary to ensure correctness. The program uses mutexes to protect critical sections and ensure thread-safe operations.

Structures Used
- *transferOperation*: Contains details about a transfer, including serial number, amount, source account, and destination account.
- *Account*: Contains the account ID, initial and current balance, a mutex to protect balance updates, and a log of operations performed on the account.

Synchronization
- *Account Mutex*: Each account has a dedicated mutex to protect its balance from concurrent modifications. This ensures that when multiple threads try to update the same account, they do so in a controlled manner, preventing data races.
- *Global Check Mutex*: A global mutex (checkMutex) is used to protect the shared boolean variable checkPassed. This variable is used during periodic consistency checks by the threads to ensure no account's balance is corrupted.

Test Platform:
CPU: 12th Gen Intel(R) Core(TM) i7-12700H 2.70 GHz
RAM: 16 GB
Operating System: Windows 11

Tests Conducted:
Operations: 100,000
Threads: 1
Time: 7.13149 seconds

Operations: 100,000
Threads: 2
Time: 3.65493 seconds

Operations: 100,000
Threads: 4
Time: 2.10215 seconds

Operations: 100,000
Threads: 8
Time: 1.27097 seconds