Parallel and Distributed Programming
Lab 1
Pop Daniel Avram (936)

Assignment: **Problem 2** - Bank Accounts
Hardware:

```
dani@dani:~$ sudo lshw -short
[sudo] password for dani:
H/W path              Device     Class          Description
=========================================================
                                 system         SATELLITE L50-C (PSKWTE)
/0                               bus            Type2 - Board Product Name1
/0/0                             memory         128KiB BIOS
/0/4                             processor      Intel(R) Core(TM) i7-5500U CPU
/0/4/8                           memory         32KiB L1 cache
/0/4/9                           memory         256KiB L2 cache
/0/4/a                           memory         4MiB L3 cache
/0/7                             memory         32KiB L1 cache
/0/27                            memory         12GiB System Memory
/0/27/0                          memory         8GiB SODIMM DDR3 Synchronous 16
/0/27/1                          memory         4GiB SODIMM DDR3 Synchronous 16
```

Multithreading policy: I have used locking on individual accounts while making operations on them, considering this a way better option than locking the whole bank for one single transaction. The strategy to avoid deadlocks was suggested by colleague Alex-Ovidiu Popa (group 936) and it implies locking the Account threads in the natural order of their UIDs. I have also used a checker thread that wakes up at each 0.1s. While auditing accounts, it holds a lock on them.
Tests: (All tests were done on a number of 10000 accounts to avoid random collisions of account generation as much as possible)

1. **5** Threads & **5000** Transactions -> 0.26s
2. **5** Threads & **50000** Transactions -> 1.06s
3. **5** Threads & **500000** Transactions -> 9.01s
4. **10** Threads & **5000** Transactions -> 0.17s
5. **10** Threads & **50000** Transactions -> 0.92s
6. **10** Threads & **50000** Transactions -> 8.45s
7. **20** Threads & **5000** Transactions -> 0.15s
8. **20** Threads & **50000** Transactions -> 0.89s
9. **20** Threads & **500000** Transactions -> 7.2s