DBMS HOMEWORK #2

Important Note:
At the begining of your answer sheet, **note your Postgresql version** that you used for this homework.
At the answers, **show your work**.
You are expected to work **in a group of at most 2.**

AIM:
To see
"Read committed" runs faster, but results with incorrect answers
"Serializable" results with always correct, but lower throughput.

TEST ENVIRONMENT:
Our company's accounting office is trying to pay to its employees their monthly payments.
To simplify the problem assume that the database has an ACCOUNTS(accno integer, balance integer) table that holds 101 rows. While the account-0 has initially 100 lira, and all others numbered as 1,2,...100 has 0 lira. We are sending 1 lira from account-0 to each other accounts (account-1 to 100) and eventually expecting to see 0 lira at account-0 and 1 lira at each other. This can be accomplished in different forms, one of which is described below.

```
TX A:
---------
e <-- SELECT balance FROM ACCOUNTS WHERE account=i;
UPDATE Accounts SET balance=e+1 WHERE account=i;
c <-- SELECT balance FROM Accounts WHERE account=0;
UPDATE Accounts SET balance=c−1 WHERE account=0;
```

For each employee ($1 \le i \le 100$) run the TX-A with isolation level **READ COMMITTED**.
Execute a number of TXs concurrently, ranging from 1 to 5.

- For the correctness, measure a *c-value*, "c1-c2/100" where c1 is the balances of account 0 before and after running all transactions, respectively. (*c-value* =1 means "totally correct")
- For throughput (Tx/sec,=*TPS*), first calculate complete execution latency,t, then find *TPS* as 100/t.

Plot correctness *c-value and TPS* for different experiments (i.e. # of concurrent TX, 1 to 5.)

Repeat experiments above for **SERIALIZABLE** isolation level and plot the corresponding graphs.

Compare your results briefly.

REFERENCES:
----
https://www.postgresqltutorial.com/postgresql-python/transaction/
https://pynative.com/python-postgresql-transaction-management-using-commit-and-rollback/
https://www.postgresql.org/docs/current/transaction-iso.html