CS 387 - Lab 4 Inlab - Accessing Postgres programmatically

So far, you have used PSQL and PGADMIN to access postgresql. Eventually, you have to access databases from programming languages to build real applications. This exercise helps you understand how to connect to postgres and execute queries from Python. You can do something similar from Java, C++ and so on through a connector called JDBC/ODBC which is a standard way of accessing relational DBs - however, we will leave it to you to experiment with this.

Exercise-1: Connecting to Postgres and executing a SQL query

For this exercise, we have provided you a zip file **(rollnumber-a4.zip)** containing **config.py**, **exec.py, ipl_ddl.sql** and **ipl_data.sql** (same data from the lab3). Starting with this, you must do the following:

1. Create a database named **lab4db** in postgres using the steps mentioned in inlab2
2. Load **ipl_ddl.sql** and **ipl_data.sql** files we have supplied in the zip file.
3. Change postgres user password. Steps below
    a. psql -U postgres -h 127.0.0.1 -p 5432 (or simply psql)
    b. \password
    c. Enter the new password twice. You will be needing this password for connecting to postgres server from python
4. You will be storing all the user-specific data in a separate configuration file. Update the **`config.py`** file (if necessary) with the appropriate database name, username, password, host, and port number
5. Fill in the TODOs in the file **`exec.py`**. This program takes a simple sql query as an argument and executes it. Look below for the sample execution
   For documentation, refer: https://www.python.org/dev/peps/pep-0249/

```
~$ python3 exec.py 'select * from venue;'
venue_id,venue_name,city_name,country_name
1,MA Chidambaram Stadium, Chepauk,Chennai,India
2,Rajiv Gandhi International Stadium, Uppal,Hyderabad,India
.
.
.
16 16,Holkar Cricket Stadium,Indore,India
~$
```

Exercise-2(Random Oracle): Issuing DDL and DML commands from Python

In the previous exercise, you've learned how to connect to postgres and run simple queries using python. In this exercise, you will create and populate a table, fetch rows from the table using a python client.

In cryptography, a **random oracle** is an oracle (a theoretical black box) that responds to every unique query with a ***random response chosen uniformly from its output domain***. ***If a query is repeated, it responds the same way every time that query is submitted***.

In this exercise, you have to create a table named **`oracle`** (in the **`lab4db`** database) with two columns, one for the query parameters (integer) and the other for the output (256-bit hex string) **using python** (including table creation). The query parameter's range is [0, 2^{16}) and the output must be a **256-bit hexadecimal string.** Your python code should print the output for every parameter value requested. You can use the same **`config.py`** file from Exercise 1 to connect to the database. Your program should be in a file **oracle.py.** Here are some sample runs of the program.

```
~$  python3 oracle.py
Query:
10
1c04fb12c0f5074cb8261b83686b02d9f0db99fa1ff3144671ca2f6446c9a2e9
Query:
3
6d61009c7f585fee977a5a7ff04776cc5368a26c17af9e96ced3bb59869011c1
Query:
10
1c04fb12c0f5074cb8261b83686b02d9f0db99fa1ff3144671ca2f6446c9a2e9
.
.
.
Query:
-1
exiting...
~$
```

Submission Instructions

- Make a folder named **<yourrollnumber>-a4** containing **config**.**py**, **exec.py** and **oracle.py**.
- Your submission should not contain any other files
- **Zip** the folder and upload it to the **moodle**
- Your code **must work** on the **docker image that we had put out.**

- Before submitting, check all the file names and run your code one more time to ensure that everything is working fine as we'll only run the python files described above via a script and ANY naming errors will result in you not receiving any marks.

Grading Rubric

Exercise	Marks
1	7.5(for each TODO) * 2 = 15
2	3(oracle table creation) + 7 (remaining task) = 10
Total	25