End Term Project: Data Synchronization Across Heterogeneous Systems

In this project, you are expected to demonstrate your understanding of data integration and synchronization across heterogeneous systems. Building on the concepts explored in Assignment 3, assume that the file `GradeRosterReport.csv` has been redundantly loaded onto at least **three heterogeneous data systems**. The systems must include **MongoDB** (a NoSQL document store) and any two or more from the following: **Pig**, **Hive**, or **PostgreSQL/MySQL**.

You are required to implement methods for reading and updating data independently in each of these systems. Furthermore, each system should define an abstract function named `merge()`, which takes as an argument the name of another system whose state is to be merged with the calling system. The `merge()` function should not directly access the data from the other system; instead, it must rely solely on an **operation log (oplog)** that records the sequence of operations performed in that system. The operation log should be designed in a way that it is generic and applicable to any table, not just `GradeRosterReport`.

As a first step, identify the possible CRUD operations supported by each chosen system (e.g., MongoDB, Pig, Hive, PostgreSQL/MySQL) and define them as the services offered by each system. From these, **consider only the operations relevant to reading and updating** the `Obtained Marks/Grade` field for a given student ID.

Design the merge functionality specific to each system. For example, a command like `MERGE(PIG, POSTGRESQL)` should merge the current state of the Pig table with the state of the PostgreSQL table (but not vice versa). The merge should be performed based on the operation logs of the two systems involved. These merge commands will be provided as input to your main program, which should execute them in the given order to synchronize the systems.

Discuss the mathematical properties of the merge operations, such as *associativity*, *commutativity*, and *idempotency*. Reflect on how convergence would occur in these scenarios, given the types of operations each system supports.

Define the structure of operation logs (oplogs) for each system using the following example formats:

Sample Operation Logs

oplog.hiveql

```
1, INSERT(4850407845c94b6a035e, A)
2, READ(4850407845c94b6a035e)
```

oplog.sql (PostgreSQL or MySQL)

```
1, READ(373807845c94b6a0335e)
2, READ(4850407845c94b6a035e)
```