Input File Format

The test input file is a CSV file and follows the format described in **Section 6.4** of the lab description. The server IDs are `[S1, S2, S3, S4, S5, S6, S7, S8, S9]` and Client records have IDs ranging from `[1, 2, 3, ... , 3000]`. These will remain consistent across all test cases.

Example Test Cases

Set Number	Transactions	Live Servers	Contact Servers
1	(21, 700, 1)	[S1, S2, S3, S4, S5, S6, S7, S8, S9]	[S1, S4, S7]
	(1301, 1302, 3)		
	(2001, 2650, 7)		
2	(301, 1302, 9)	[S1, S3, S4, S5, S7, S8, S9]	[S1, S4, S7]
	(501, 502, 3)		

Table 1: Example Test Cases

Transaction Processing

- **Initial State:** Each client record starts with an initial balance of 10 units. Each server maintains all the client balances in the datastore.
- **Transaction Handling:** Your client must process and execute transactions in the order they are listed in the input file. For each transaction, the client determines the corresponding shard based on the Shard Mapping and sends the transaction to the Contact server of the appropriate cluster.

Explanation:

- The first set (Set Number 1) contains three transactions `(21, 700, 1)`, `(2001, 2650, 7)` and `(1301, 1302, 3)`. These transactions are processed with all servers active, as indicated by the list `[S1, S2, S3, S4, S5, S6, S7, S8, S9]`. In this setup, S1 serves as the contact server for Cluster C1, S4 for Cluster C2, and S7 for Cluster C3.
- The second set (Set Number 2) contains two transactions `(301, 1901, 9)` and `(501, 502, 3)`. For these transactions, only the servers `[S1, S3, S4, S5, S6, S7, S9]` are active, indicating that servers S2 from Cluster C1 and S6 from Cluster C2 are **disconnected**. As in the previous set, S1 continues to serve as the contact server for Cluster C1, S4 for Cluster C2, and S7 for Cluster C3.

Refer to **Section 6.4** in the lab description for detailed instructions on running the test cases from the input file.

NOTE

- Make sure your client does not process the next set of transactions unless explicitly instructed using the user prompt.
- Keep in mind that test case sets may be interdependent, meaning that server states and the datastore are carried forward from one set to the next.
- Furthermore, your implementation must strictly expose the functions outlined in Section 2.2 of the lab description, as these functions will be used during this period to verify the state of your servers.

We are particularly interested in the output of the functions described in Section 2 of the lab description, particularly the `PrintBalance()` and `PrintDatastore()` functions.

The expected output of the `PrintBalance(1302)` function, after completing the execution of both Set 1 and Set 2 is displayed below:

Server	**1302**
S4	22
S5	22
S6	13

Table 2: Expected Output of `PrintBalance(1302)` after Executing Set 1 and Set 2

NOTE: Even if a server is disconnected, the balance of the specified Client ID must still be displayed for that server when requested using the `PrintBalance()` function. Similarly, the datastore of all servers should be shown, regardless of their connection status.

The expected output of the `PrintDatastore()` function, after completing the execution of both Set 1 and Set 2 is shown below. Each entry in the datastore consists of (1) a ballot number, (2) a character 'P' or 'C' if the entry is for a cross-shard transaction and (3) the actual transaction.

Server	**Datastore**
S1	[<1,1>, (21, 700, 1)] → [<2,1>, P, (301, 1302, 9)] → [<3,1>, (501, 502, 3)] → [<2,1>, C, (301, 1302, 9)]
S2	[<1,1>, (21, 700, 1)]
S3	[<1,1>, (21, 700, 1)] → [<2,1>, P, (301, 1302, 9)] → [<3,1>, (501, 502, 3)] → [<2,1>, C, (301, 1302, 9)]
S4	[<1,4>, (1301, 1302, 3)] → [<2,4>, P, (301, 1302, 9)] → [<2,4>, C, (301, 1302, 9)]
S5	[<1,4>, (1301, 1302, 3)] → [<2,4>, P, (301, 1302, 9)] → [<2,4>, C, (301, 1302, 9)]
S6	[<1,4>, (1301, 1302, 3)]
S7	[<1,7>, (2001, 2650, 7)]
S8	[<1,7>, (2001, 2650, 7)]
S9	[<1,7>, (2001, 2650, 7)]

Table 3: Expected Output of `PrintDatastore()` after Executing Set 1 and Set 2

Test Case Descriptions

- Test 1: Concurrent transaction in multiple clusters commit - Intra Shard
- Test 2: Concurrent transaction in same clusters commit - Intra Shard
- Test 3: Abort due to insufficient balance - Intra Shard
- Test 4: Abort due to insufficient balance and disconnected servers - Intra Shard
- Test 5: Abort due to no majority - Intra Shard
- Test 6: Concurrent transaction commit - Cross Shard
- Test 7: Abort due to unavailability of lock - Cross Shard (Part 1)
- Test 8: Abort due to unavailability of lock - Cross Shard (Part 2)
- Test 9: Abort and rollback WAL due to insufficient balances - Cross Shard (Part 1)
- Test 10: Abort and rollback WAL due to insufficient balances - Cross Shard (Part 2)

The descriptions provided above outline the test cases from the CSV input file. These test cases aim to cover all major scenarios and should be sufficient for validating your implementation.