Explaining the server code:

1. The connectclient() sets the connection socket. Then it checks if the incoming request is a GET or STORE. If it is a GET, it fetches the corresponding value for the passed KEY from the dictionary.
2. If it is a STORE, the passed KEY-VALUE pair is stored in the dictionary. If the key already exists, the value for it is overwritten (I assume that the client wants to do an UPDATE if an existing key is sent).
3. I added an additional keyword QUIT for the server to know when to stop listening to the client requests. Whenever the request has QUIT in it, the server gracefully terminates.
4. The writedata() is called when the server gets a STORE key-value request. This function acquires a lock on the current object of Server class and writes data to the dictionary.

Explaining the client code:

The runalice() or runbob() function takes parameterized input in the form :

```
Python3 alice.py --key=foo --value=foo-value
```

It parses the input and prepares a GET, STORE or QUIT request for the server. This request is then sent to the server using sendall()

Explaining the driver code:

1. The driver code first forks a process and calls the server.py. It then forks tow another processes and calls alice.py and bob.py.
2. Then alice and bob execute various requests to the client.
3. At the end bob sends a QUIT request to the server and the server terminates.

Execution:

Place all 4 files in the same directory and execute driver.py with python3 as follows:

```
python3 driver.py
```

Execute individual clients as follows:

```
python3 server.py
Python3 alice.py --key=foo --value=foo-val   (same applies for bob.py)
Python3 bob.py --key=foo                     (same applies for alice.py)
Python3 bob.py --key=QUIT
```

Output:

The following is the output:

Testing:

To try testing different conditions, a test case can be added to the driver. Or the individual files can be tested as shown under execution.

The current test cases in driver.py produce the following output:

```
vinitaboolchandani@Vinitas-MacBook-Pro HW1 % python3 driver.py

 Server Starting
The server is ready to receive

 TEST CASE: Alice STORE foo=foo-val-none                    -> alice stores a new key-value for foo
Server Response in Alice: b'Saved Successfully for key:foo'

 TEST CASE (INVALID KEY): Alice GET abc                     ->  alice gets a non existing value
Server Response in Alice: b'Invalid Key:abc'

 TEST CASE (CONCURRENCY): Alice STORE foo=foo-val-alice     ->alice updates value for foo while bob is
                                                              trying the same. Needed some communication
                                                              to make them do this simultaneously
Server Response in Alice: b'Update Successfully for key:foo'

 TEST CASE: Alice GET foo
Server Response in Alice: b'foo-val-alice'
vinitaboolchandani@Vinitas-MacBook-Pro HW1 %
 TEST CASE: Bob GET foo                                     ->  bob tries to get value for foo
Server Response in Bob: b'foo-val-alice'

 TEST CASE: Bob STORE foo-bob=foo-bob-val-bob               ->  bob stores a new key-value
Server Response in Bob: b'Saved Successfully for key:foo-bob'

 TEST CASE (CONCURRENCY): Bob STORE foo=foo-val-bob         ->  bob executes the same update as
                                                                alice
Server Response in Bob: b'Update Successfully for key:foo'

 Sending STOP SERVER message to Bob                         ->  Bob tells the server to quit


 Drives ends
Server Response in Bob: b'Server Quitting..'
```