Transactional Key Value Store

The assignment is to build an interactive command line interface to a transactional key value store. A user should be able to compile and run this program and get an interactive shell with a prompt where they can type commands. The user can enter commands to set/get/delete key/value pairs and count values. All values can be treated as strings, no need to differentiate by type. The key/value data only needs to exist in memory for the session, it does not need to be written to disk.

The interface should also allow the user to perform operations in transactions, which allows the user to commit or roll back their changes to the key value store. That includes the ability to nest transactions and roll back and commit within nested transactions. The solution shouldn't depend on any third party libraries. The interface should support the following commands:

Supported Commands

```
SET <key> <value> // store the value for key
GET <key>         // return the current value for key
DELETE <key>      // remove the entry for key
COUNT <value>     // return the number of keys that have the given value
BEGIN             // start a new transaction
COMMIT            // complete the current transaction
ROLLBACK          // revert to state prior to BEGIN call
```

Examples

Set and get a value:

```
SET foo 123
GET foo
=> 123
```

Delete a value:

```
DELETE foo
GET foo
=> key not set
```

Count the number of occurrences of a value:

```
SET foo 123
SET bar 456
SET baz 123
COUNT 123
=> 2

COUNT 456
=> 1
```

Commit a transaction:

```
BEGIN
SET foo 123
COMMIT
GET foo
=> 123
```

Rollback an uncommitted transaction:

```
BEGIN
SET foo 123
ROLLBACK
GET foo 123
=>
```

Rollback a transaction:

```
SET foo 123
SET bar abc
BEGIN
SET foo 456
GET foo
=> 456

SET bar def
GET bar
=> def

ROLLBACK
GET foo
=> 123

GET bar
=> abc

COMMIT
=> no transaction
```

Nested transactions:

```
SET foo 123
BEGIN
SET foo 456
BEGIN
SET foo 789
GET foo
=> 789

ROLLBACK
GET foo
=> 456

ROLLBACK
GET foo
=> 123
```

Criteria and Notes

- The application commands (`GET` , `SET` , etc) should be case sensitive
- Data does not need to be saved to disk. It should only exist during the runtime of the toy application

- This application should not accept multiline input