Implement an in-memory database that has the following functions. The goal is to have a working implementation over any other aspects of the code. You should test and document as much as it's useful to you to have a working implementation. We'd rather see a working partial implementation than a non-working attempt at a complete solution. Use libraries at will (but not database-like ones or actual databases). Use Google/Stack Overflow/online references at will as well.

Imagine that it reads values from STDIN line by line and executes the functions as they happen.

The name and value will be strings with no spaces in them.

`SET [name] [value]`
: Sets the name in the database to the given value

`GET [name]`
: Prints the value for the given name. If the value is not in the database, prints `NULL`

`DELETE [name]`
: Deletes the value from the database

`COUNT [value]`
: Returns the number of names that have the given value assigned to them. If that value is not assigned anywhere, prints `0`

`END`
: Exits the database

Database must also support transactions:

`BEGIN`
: Begins a new transaction

`ROLLBACK`
: Rolls back the most recent transaction. If there is no transaction to rollback, prints `TRANSACTION NOT FOUND`

`COMMIT`
: Commits *all* of the open transactions

The number of transactions is small relative to the number of items in the database. Also, the number of actions in a transaction will be relatively small. `GET`, `SET`, `DELETE`, and `COUNT` should all have a runtime of less than $O(log\ n)$, if not better (where n is the number of items in the database). You should be responsible with space, e.g., the entire database shouldn't be doubled for every transaction.

Example #1

```
>> GET a
NULL
>> SET a foo
>> SET b foo
>> COUNT foo
2
>> COUNT bar
0
>> DELETE a
>> COUNT foo
1
>> SET b baz
>> COUNT foo
0
>> GET b
baz
>> GET B
NULL
>> END
```

Example #2

```
>> SET a foo
>> SET a foo
>> COUNT foo
1
>> GET a
foo
>> DELETE a
>> GET a
NULL
>> COUNT foo
0
>> END
```

Example #3

```
>> BEGIN
>> SET a foo
>> GET a
foo
>> BEGIN
>> SET a bar
>> GET a
bar
>> ROLLBACK
>> GET a
foo
>> ROLLBACK
>> GET a
NULL
>> END
```

Example #4

```
>> SET a foo
>> SET b baz
>> BEGIN
>> GET a
foo
>> SET a bar
>> COUNT bar
1
>> BEGIN
>> COUNT bar
1
>> DELETE a
>> GET a
NULL
>> COUNT bar
0
>> ROLLBACK
>> GET a
bar
>> COUNT bar
1
>> COMMIT
>> GET a
bar
>> GET b
baz
>> END
```