**Email:**

**Phone #:**

**Wireless network:** Box Visitor

**Username:** Registered email address

**Password:** Full name, lower-case, no spaces

This two-hour session will be organized as such: 15 minutes for review of the problem statement with interviewer present, the next 75 minutes to code a solution on your own, and the remaining 30 minutes for code review and discussion with your interviewer.

Your task is to create a very simple, **single-threaded**, **in-memory** database, which has a limited set of data and transaction commands.

**Data commands**

Your database should accept the following commands:

SET [name] [value]: Set a variable [name] to the value [value]. Neither variable names nor values will ever contain spaces.

GET [name]: Print out the value stored under the variable [name]. Print NULL if that variable name hasn't been set.

DELETE [name]: Delete the variable [name]

COUNT [value]: Return the number of variables equal to [value]. If no values are equal, this should output 0.

**Examples**

So here is a sample input and output:

>>> SET a 10

>>> GET a

10

>>> DELETE a

>>> GET a

NULL

And another one:

>>> SET a 10

>>> SET b 10

>>> COUNT 10

2

>>> COUNT 20

0

>>> DELETE a

>>> COUNT 10

1

>>> SET b 30

>>> COUNT 10

0

**Transaction commands**

In addition to the data commands, your database should support transactions, accepting the following commands:

BEGIN: Open a transactional block

ROLLBACK: Rollback all of the commands from the *most recent*transactional block.

COMMIT: Permanently store all of the operations from *all* presently open transactional blocks.

Both ROLLBACK and COMMIT cause the program to print 'NO TRANSACTION' if there are no open transaction blocks.

Your database needs to support nested transactions. ROLLBACK only applies to the most recent transaction block, but COMMIT applies to all transaction blocks. (Any data command run outside of a transaction is committed immediately.)

**Examples**

Input/Output:

>>> BEGIN

>>> SET a 10

>>> GET a

10

>>> BEGIN

>>> SET a 20

>>> GET a

20

>>> ROLLBACK

>>> GET a

10

>>> ROLLBACK

>>> GET a

NULL

Input/Output:

>>> BEGIN

>>> SET a 30

>>> BEGIN

>>> SET a 40

>>> COMMIT

>>> GET a

40

>>> ROLLBACK

NO TRANSACTION

Input/Output:

>>> SET a 50

>>> BEGIN

>>> GET a

50

>>> SET a 60

>>> BEGIN

>>> DELETE a

>>> GET a

NULL

>>> ROLLBACK

>>> GET a

60

>>> COMMIT

>>> GET a

60

Input/Output:

>>> SET a 10

>>> BEGIN

>>> COUNT 10

1

>>> BEGIN

>>> DELETE a

>>> COUNT 10

0

>>> ROLLBACK

>>> COUNT 10

1

**Input Processing**

Commands should be processed one at a time.

You have the freedom to choose how your program consumes input. Using stdin, or function calls to invoke each of the commands, or driving them via unit tests is perfectly fine. Choose whichever is easiest for you.

**Time Complexity**

When thinking about the structure of your database, assume that the most common operations are GET, SET, DELETE, and COUNT, all of which are equally common. All these commands should have worst-case time complexity of O(log(N)) or better, where N is the total number of variables stored in the database. Note, N does not depend on the number of transactions, T. For example, when N=1, T can be arbitrarily large.

**Space Complexity**

Your solution should be efficient about how much memory is used by a transaction (i.e. it should not nearly double your program's memory usage every time a BEGIN is called). You can assume that only a small number of values will be changed in a transaction.