In memory database

Implement an in-memory database similar to Redis. Your program should be able to read commands via standard input (stdin), and should print appropriate responses to standard output (stdout).

Guidelines

Please **ONLY** use either Javascript,Typescript, Python, Java, PHP, Go or Ruby for your implementation. We're much more interested in seeing clean code, solid program design and good algorithmic performance than raw throughput. Please reduce the use of external dependencies and include instructions on how to run your code directly from the command line, without the use of an IDE.

Data Commands

Your database should accept the following commands:

SET name value – Set the variable *name* to the value *value*. Neither variable names nor values will contain spaces.

GET name – Print out the value of the variable *name*, or NULL if that variable is not set.

UNSET name – Unset the variable *name*, making it just like that variable was never set.

NUMEQUALTO value – Print out the number of variables that are currently set to *value*. If no variables equal that value, print 0.

END – Exit the program. Your program will always receive this as its last command. Commands will be fed to your program one at a time, with each command on its own line. Any output that your program generates should end with a newline character.

Input	Output
SET x 10	
GET x	10
UNSET x	
GET x	NULL
END	

Input	Output
SET a 10	
SET b 10	
NUMEQUALTO 10	2
NUMEQUALTO 20	0
SET b 30	
NUMEQUALTO 10	1
END	

Transaction Commands

In addition to the above data commands, your program should also support database transactions by also implementing these commands:

BEGIN – Open a new transaction block. Transaction blocks can be nested; a BEGIN can be issued inside of an existing block.

ROLLBACK – Undo all of the commands issued in the most recent transaction block, and close the block. Print nothing if successful, or print **NO TRANSACTION** if no transaction is in progress.

COMMIT – Close all open transaction blocks, permanently applying the changes made in them. Print nothing if successful, or print **NO TRANSACTION** if no transaction is in progress.

Any data command that is run outside of a transaction block should commit immediately. Here are some example command sequences:

Input	Output
BEGIN	
SET a 10	
GET a	10
BEGIN	
SET a 20	

GET a	20
ROLLBACK	
GET a	10
ROLLBACK	
GET a	NULL
END	

Input	Output
BEGIN	
SET a 30	
BEGIN	
SET a 40	
COMMIT	
GET a	40
ROLLBACK	NO TRANSACTION
END	

Input	Output
SET a 50	
BEGIN	
GET a	50
SET a 60	
BEGIN	
UNSET a	
GET a	NULL

ROLLBACK	
GET a	60
COMMIT	
GET a	60
END	

Input	Output
SET a 10	
BEGIN	
NUMEQUALTO 10	1
BEGIN	
UNSET a	
NUMEQUALTO 10	0
ROLLBACK	
NUMEQUALTO 10	1
COMMIT	
END	

Performance Considerations

The most common operations are **GET**, **SET**, **UNSET**, and **NUMEQUALTO**. All of these commands should have an expected worst-case runtime of O(log N) or better, where N is the total number of variables stored in the database. The vast majority of transactions will only update a small number of variables. Accordingly, your solution should be efficient about how much memory each transaction uses.

Time limit: **3 hours**