Work Sample Simulation Overview

K/V REPL with nested transactions

In this exercise we ask you to write a command line REPL (read-eval-print loop) that drives a simple in-memory key/value storage system. This system should also allow for nested transactions. A transaction can then be committed or aborted.

Candidate Instructions

Use whatever programming language, tools, and development environment you're most comfortable with. Be prepared to talk about your submission; you and one of our engineers will be walking through it.

For simplicity, all keys and values are simple ASCII strings delimited by whitespace. No quoting is needed.

All errors are output to stderr.

Commands are case-insensitive.

As this is a simple command line program with no networking, there is only one "client" at a time. There is no need for locking or multiple threads.

Example Run

```
$ my-program
> WRITE a hello
> READ a
hello
> START
> WRITE a hello-again
> READ a
hello-again
> START
> DELETE a
> READ a
Key not found: a
> COMMIT
> READ a
Key not found: a
> WRITE a once-more
> READ a
once-more
> ABORT
> READ a
hello
> QUIT
Exiting...
```

Commands

- **READ <key>** Reads and prints, to stdout, the *val* associated with *key*. If the value is not present an error is printed to stderr.
- **WRITE <key> <val>** Stores *val* in *key*.
- **DELETE <key>** Removes all *key* from store. Future *READ* commands on that *key* will return an error.
- **START** Start a transaction.

- **COMMIT** Commit a transaction. All actions in the current transaction are committed to the parent transaction or the root store. If there is no current transaction an error is output to stderr.
- **ABORT** Abort a transaction. All actions in the current transaction are discarded.
- **QUIT** Exit the REPL cleanly. A message to stderr may be output.

Work Sample Simulation Interview Structure and Details

Schedule:

- 5 minutes: Introductions
- 25 minutes: Discuss code and assumptions/alternatives with engineer
- 15 minutes: Q&A

Format:

- In person presenation