COMPILERS QUIZ 2

In this exercise, we will design a Forth-like language. This language has the following elements:

- Numbers (with decimal point): 42, 42.5.
- Strings: "Hello, World!"
- Words: Everything else without spaces. Example: get, put, +. The following are not words: 2a, "aa.

A program is a sequence of elements separated by whitespace. The language has a postfix syntax and uses a stack for evaluation. The following program prints 5 to screen.

```
,----
| 2 3 + put
`----
```

Numbers or strings are pushed to the stack. Words correspond to operations. They take arguments by popping the stack and pushes result back into stack.

The following reads two numbers and prints their sum.

```
,----
| get get + put
`----
```

For every k-ary operator, the top element is the last argument, the one below top is the second-last argument and so on. The following prints 5.

```
,----
| 10 2 / put
`----
```

We will denote a stack S by x y ... where x is the top element. We use x S to denote a stack where x is the top element with some arbitrary elements S under it.

Implement an interpreter with the following words:
- +, -, *, /, ^ for arithmetic.
- get for reading a value (number or string from input) from keyboard. The read value is pushed to stack.
- put for printing a value (followed by newline) to screen.
- pop for removing top element from stack.
- dup for turning stack from x S to x x S.
- rot for turning stack from x y S to y x S.
- concat for concatenating two strings.

Here's a program that queries two numbers from input and outputs their sum.

```
,----
| "Enter first number: "  put get
| "Enter second number: " put get
| "Their sum is: " put + put
`----
```