Arithmetic expression evaluation

Goal. Evaluate infix expressions.

```
( 1 + ( ( 2 + 3 ) * ( 4 * 5 ) ) )
```

operand operator

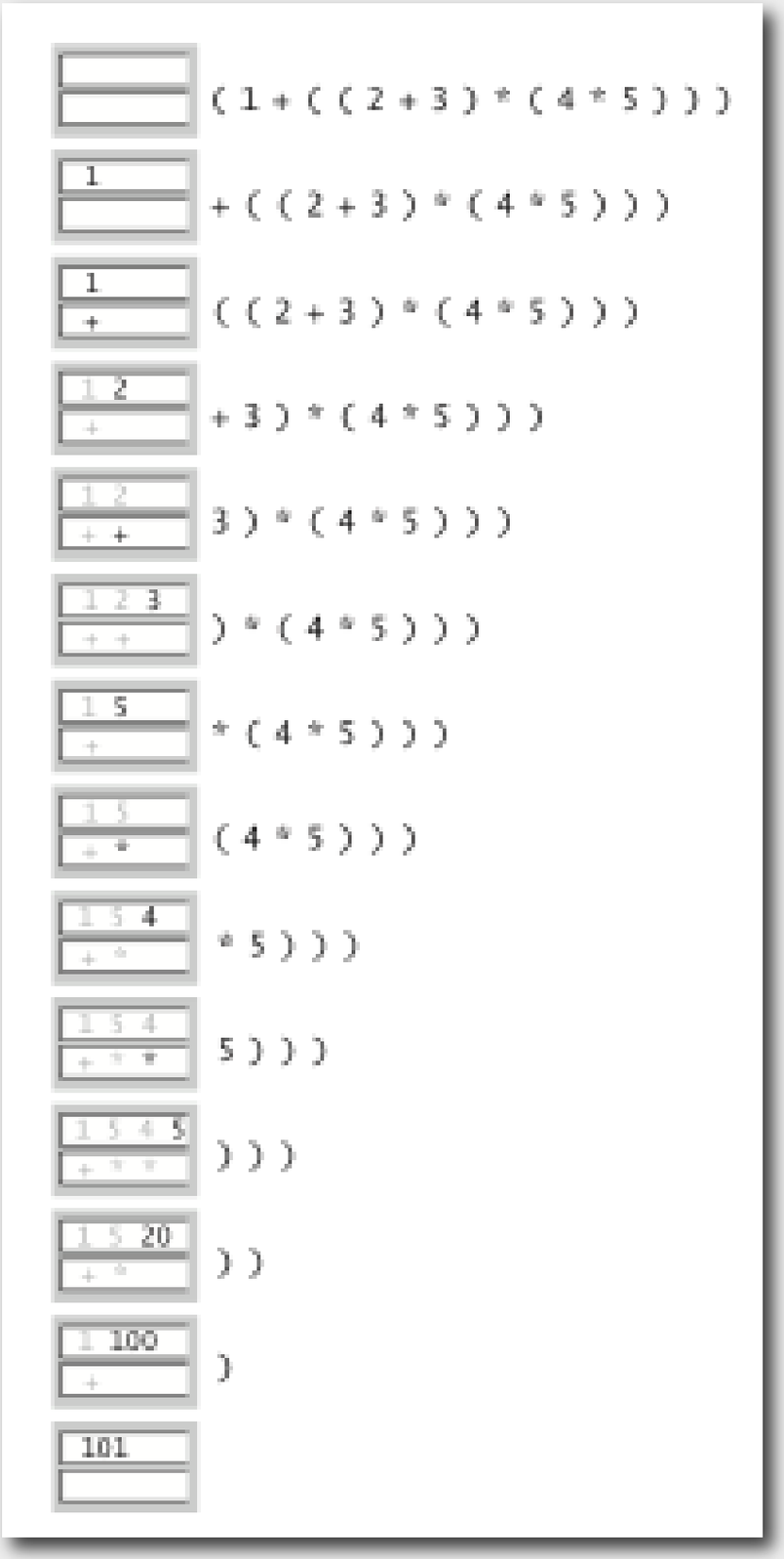
Two-stack algorithm. [E. W. Dijkstra]

- Value: push onto the value stack.
- Operator: push onto the operator stack.
- Left parenthesis: ignore.
- Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the operand stack.

Context. An interpreter!

LET THE GAMES BEGIN!!!

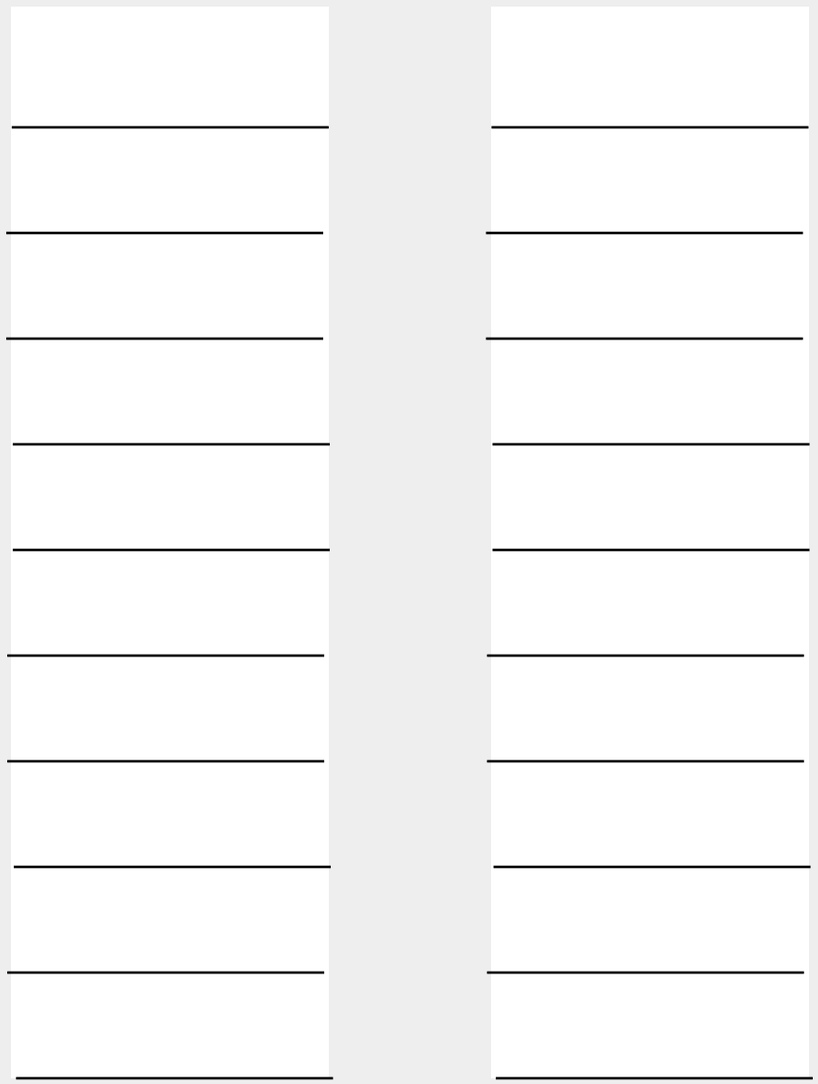
value stack
operator stack



Example Arithmetic Stack Evaluation (TEAM 1+2+3+4)

Expression:

(4 + 3) = ?



Value Stack

Operator Stack

Example Arithmetic Stack Evaluation (TEAM 1+2)

Expression:

(((1 + 3) + (8 - 4)) / ((32 - 16) / (2 - 0))) = ?

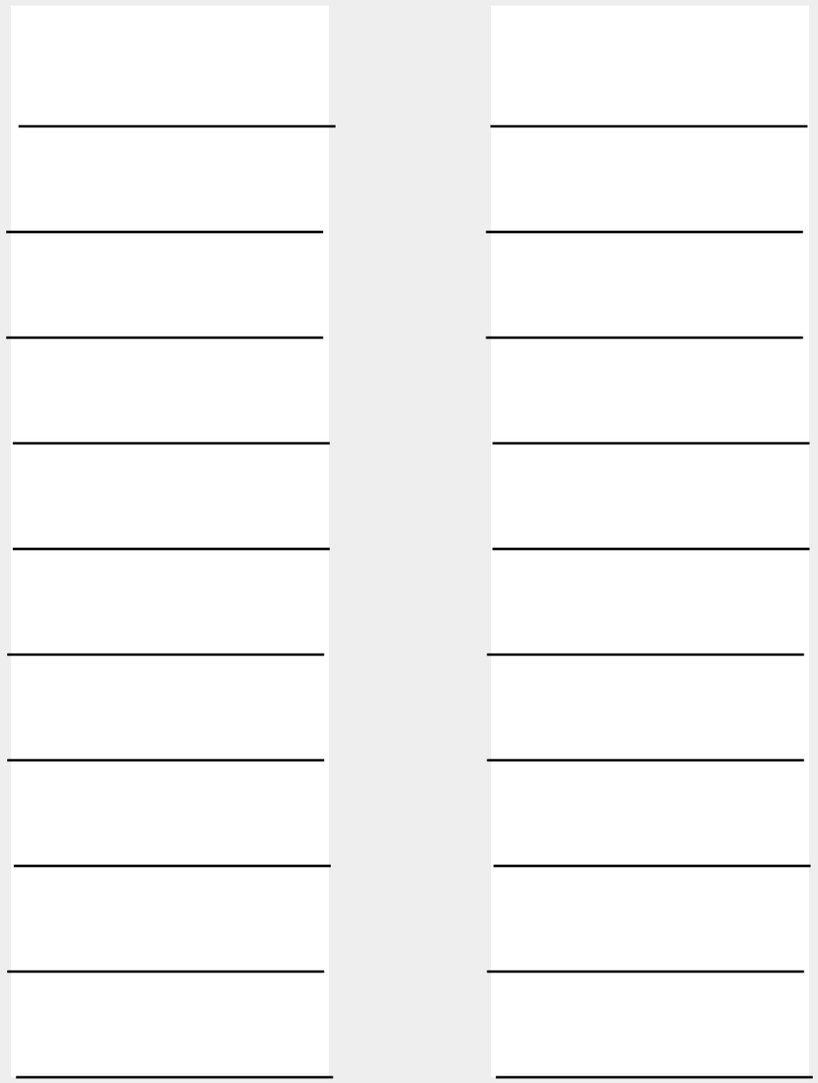
Value Stack

Operator Stack

Example Arithmetic Stack Evaluation (TEAM 3+4)

Expression:

(((5 - 1) + (2 + 2)) / ((108 - 92) / (1 + 1))) = ?



Value Stack

Operator Stack

Correctness

Q. Why does it work correctly?

A. When algorithm encounters an operator surrounded by two values within parentheses, it leaves the result on the value stack.

```
( 1 + ( ( 2 + 3 ) * ( 4 * 5 ) ) )
```

as if the original input were:

```
( 1 + ( 5 * ( 4 * 5 ) ) )
```

Repeating the argument:

```
( 1 + ( 5 * 20 ) )
( 1 + 100 )
101
```

Extensions. More ops, precedence order, associativity.

Stack-based programming languages

Observation 1. The 2-stack algorithm computes the same value if the operator occurs after the two values.

```
( 1 ( ( 2 3 + ) ( 4 5 * ) * ) + )
```

Observation 2. All of the parentheses are redundant!

```
1 2 3 + 4 5 * * +
```

Jan Lukasiewicz

Bottom line. Postfix or "reverse Polish" notation.

Applications. Postscript, Forth, calculators, Java virtual machine, ...