Arithmetic operations

numbers

+ , - , x , /

Infix notation

↓

operators appear in between operands

2+3

2+3+5

"2 + 3 x 6"

Is the answer 30 or 20?.

BODMAS ← British

BOMDAS ← American??

Inherent ambiguity in infix notation.

With BODMAS

- Multiply 3 & 6
- Add 2 and the above result.

For the "other" interpretation use brackets

$(2+3)\times 6$

- Add 2 & 3
- Multiply result with 6.

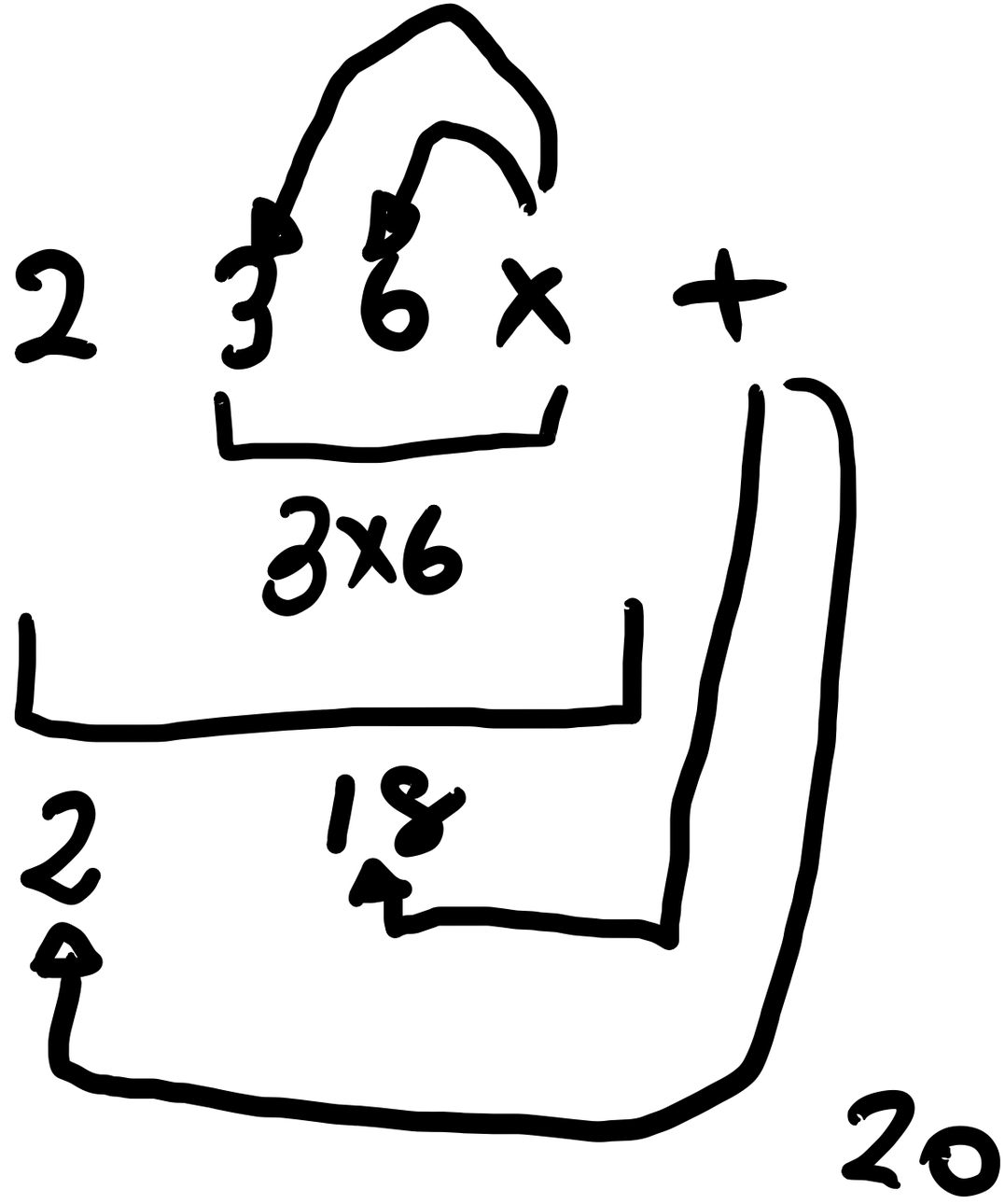
Can we represent arithmetic expressions without brackets?

Polish notation ⇔ Bracket-free!

Reverse Polish notation

2 + 3 × 6

In RPN: 2 3 6 × +



$(2+3)\times 6$

In RPN: $2\ 3\ +\ 6\ \times$

$(2+3\times(6-5)+2)/3$

In RPN:

$2\ 3\ 6\ 5\ -\ \times\ +\ 2\ +\ 3\ /$

Hw. leftmost 2+3 in brackets.

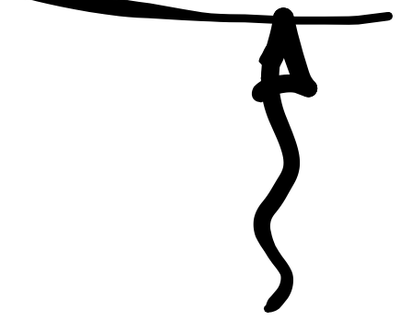
There are Programming languages based on RPN

Forth

Factor

⋮

Abstract data type : Stack



Separated from any concrete representation.

"Eleven" "ग्यारह"

11_{10} ← Because we have 10 fingers

1011_2 ← On/off

13_8

IIII IIII I

No eraser required to increment.

natural numbers

zero, one, two...

+, −, ×, / ← operations on abstract notion of a number.

Bitwise AND, OR ← operate on concrete representations of numbers

Take the last Symbol

$11 \rightsquigarrow 1$

$(13)_8 \rightsquigarrow 3$

Stack<T> ← element type

empty() → creates an empty stack

is_empty(s) → Returns true if stack is empty false o/w.

Push(s, e) → add element e to Stack s.

Pop(s) → Returns the element last added & removes it from stack.

S = empty() ↵

Pop(S) ← 5

Push(S, 5) ↵

Push(S, 2) ↵

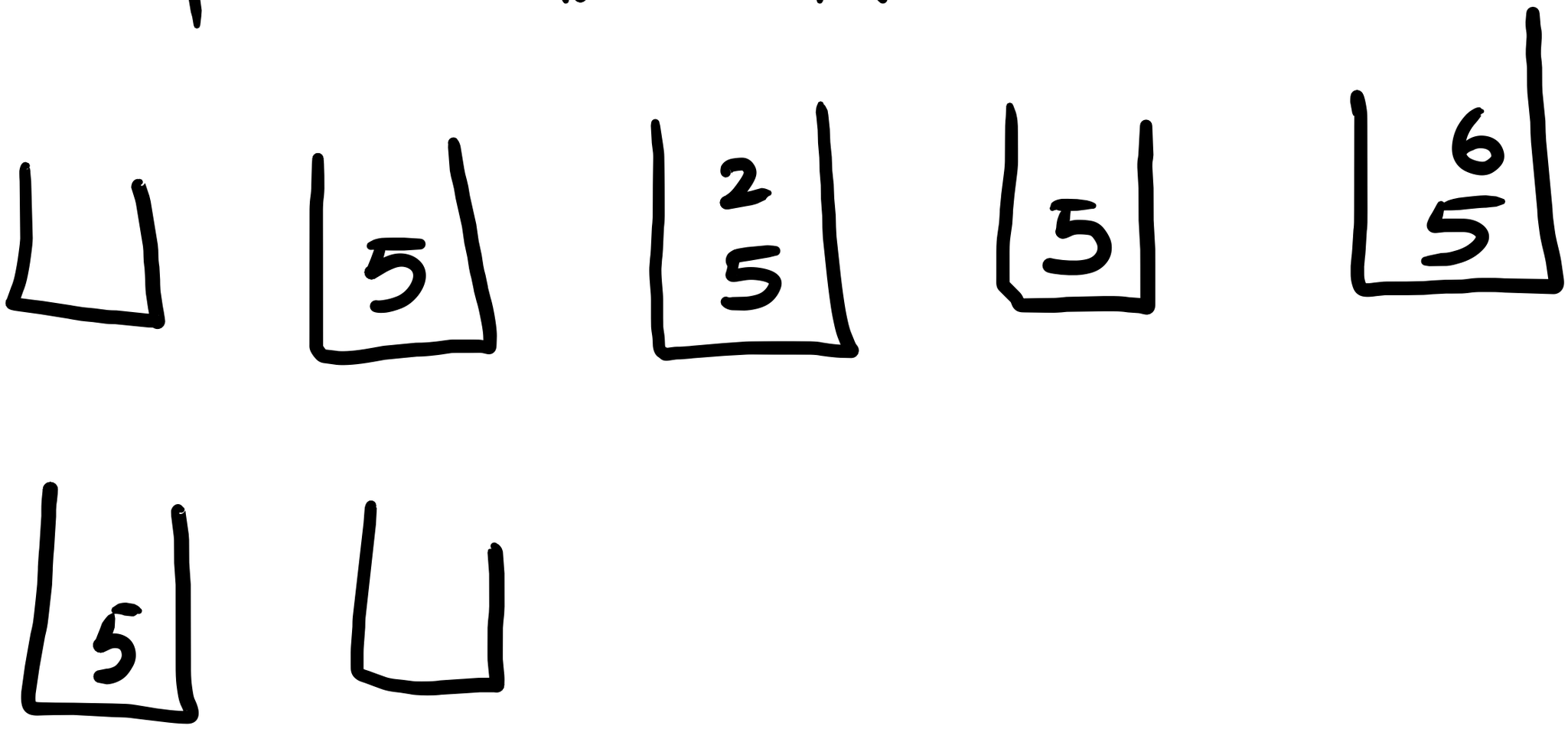
Pop(S) ← Returns 2 & removes it

Push(S, 6) ←

Pop(S) ← 6

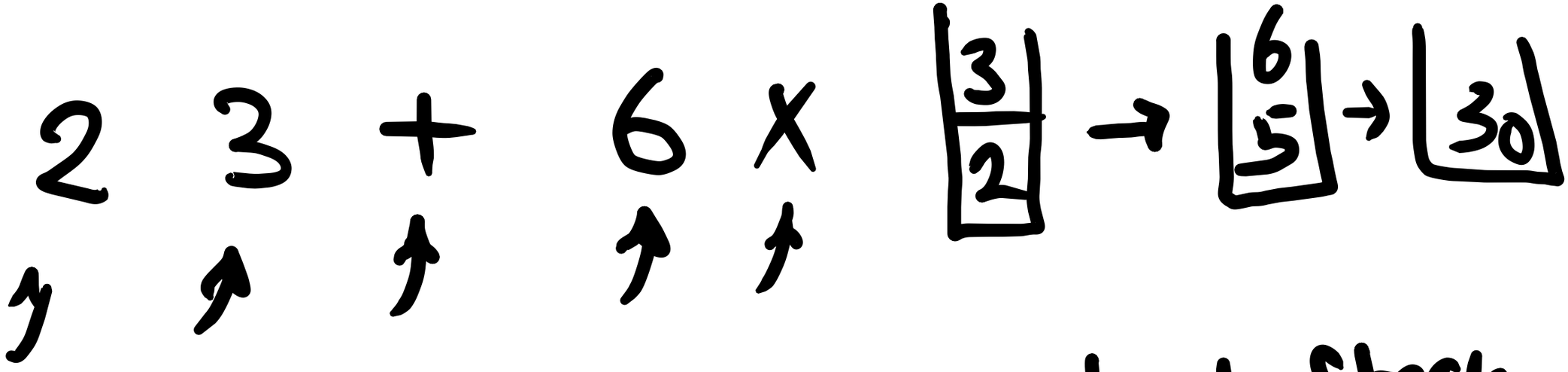
is-empty() ← NO

Why the name stack?



How is it related to RPN

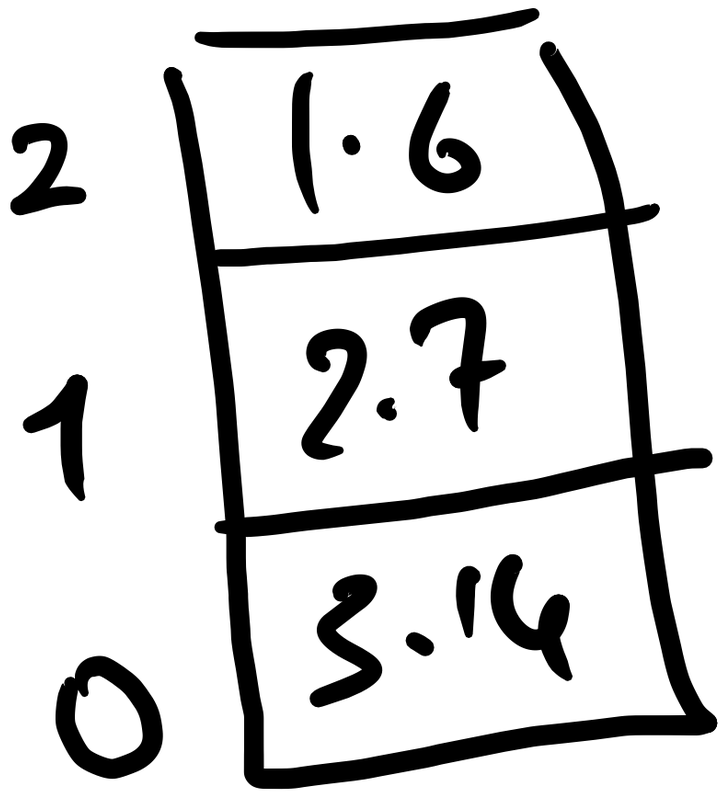
2 3 + 6 X



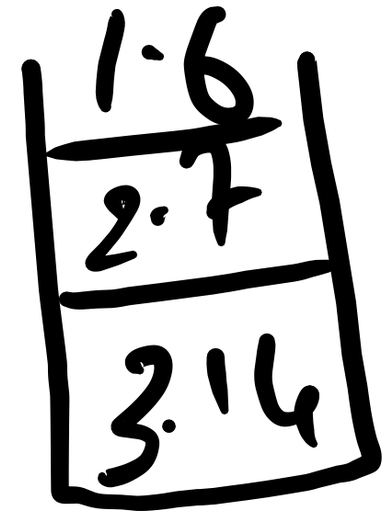
If it's a number push to Stack

If operator Pop stack to get operands
compute result & Push it.

top = 3



top=2



Suggestion