

**Goal.** Evaluate infix expressions.

\[
(1 + (2 + 3) \times (4 \times 5))
\]

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

Expression:

( 4 + 3 ) = ?

Value Stack

Operator Stack
Example Arithmetic Stack Evaluation (TEAM 1+2)

Expression:

\[
\frac{((1 + 3) + (8 - 4))}{((32 - 16) / (2 - 0))} = ?
\]
Example Arithmetic Stack Evaluation (TEAM 3+4)

Expression:

\[
\frac{( ( 5 - 1 ) + ( 2 + 2 ) )}{( ( 108 - 92 ) / ( 1 + 1 ) )} = ?
\]
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.

\[
( 1 + ( ( 2 + 3 ) \times ( 4 \times 5 ) ) )
\]

as if the original input were:

\[
( 1 + ( 5 \times ( 4 \times 5 ) ) )
\]

Repeating the argument:

\[
( 1 + ( 5 \times 20 ) )
\]
\[
( 1 + 100 )
\]
\[
101
\]

Extensions. More ops, precedence order, associativity.
Observation 1. The 2-stack algorithm computes the same value if the operator occurs after the two values.

\[
\left( 1 \left( \left( 2 \ 3 \ + \right) \left( 4 \ 5 \ * \right) \ * \right) \ + \right)
\]

Observation 2. All of the parentheses are redundant!

\[
1 \ 2 \ 3 \ + \ 4 \ 5 \ * \ * \ +
\]

Bottom line. Postfix or "reverse Polish" notation.

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