PREFIX, POSTFIX, INFIX NOTATION

Infix Notation

- To add A, B, we write

  A+B

- To multiply A, B, we write

  A*B

- The operators ('+' and '*') go in between the operands ('A' and 'B')
- This is "*Infix*" notation.

Prefix Notation

- Instead of saying "A plus B", we could say "add A,B " and write

  + A B

- "Multiply A,B" would be written

  * A B

- This is *Prefix* notation.

Postfix Notation

- Another alternative is to put the operators after the operands as in

A B +

and

A B *

- This is *Postfix* notation.

- The terms infix, prefix, and postfix tell us whether the operators go between, before, or after the operands.

Parentheses

- Evaluate 2+3*5.
- + First:

$$(2+3)*5 = 5*5 = 25$$

- * First:

$$2+(3*5) = 2+15 = 17$$

- Infix notation requires Parentheses.

What about Prefix Notation?

- + 2 * 3 5 =

  $= + 2\ \underline{*\ 3\ 5}$

  $= \underline{+\ 2\ 15} = 17$

- * + 2 3 5 =

  $= *\ \underline{+\ 2\ 3}\ 5$

  $= \underline{*\ 5\ 5} = 25$

- No parentheses needed!

Postfix Notation

- 2 3 5 * + =

  $= 2\ \underline{3\ 5\ *}\ +$

  $= \underline{2\ 15\ +} = 17$

- 2 3 + 5 * =

  $= \underline{2\ 3\ +}\ 5\ *$

  $= \underline{5\ 5\ *} = 25$

- No parentheses needed here either!

Conclusion:

- Infix is the only notation that requires parentheses in order to change the order in which the operations are done.

Precedence Rule

Please Excuse My Dear Aunt Sally

P- paranthesis

E- Exponent

M- Multiply

D- Divide

A- Addition

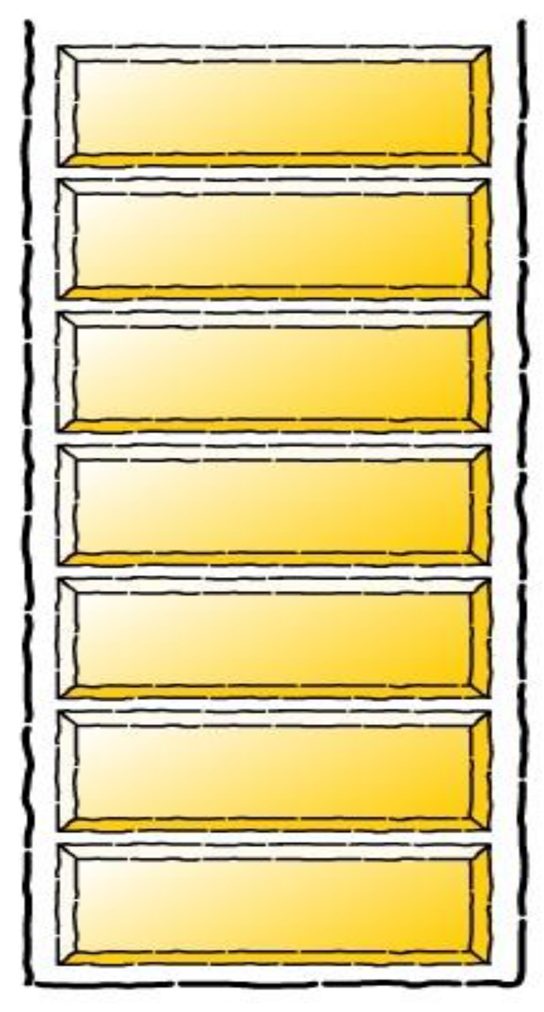
S- Subtraction.

Infix to Postfix

- Initialize a Stack for operators, output list
- Split the input into a list of tokens.
- for each token (left to right):

```
    if it is operand:  append to output
    if it is '(': push onto Stack
    if it is ')': pop & append till '('
    if it in '+-*/':
      while peek has precedence $\geq$ it:
              pop & append
      push onto Stack
pop and append the rest of the Stack.
```

Infix to postfix conversion



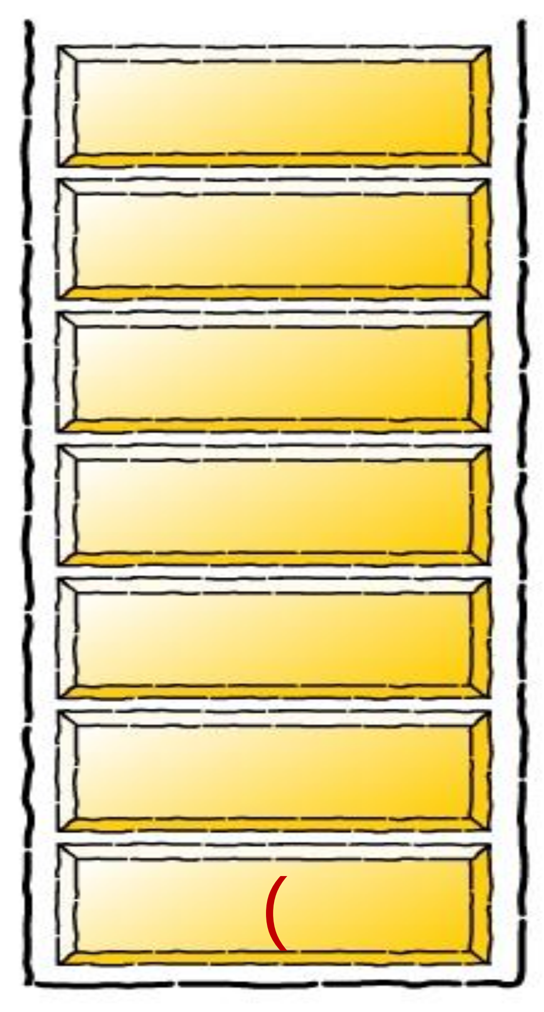
infixVect

(a + b - c) * d – (e + f)

postfixVect

Infix to postfix conversion

stackVect

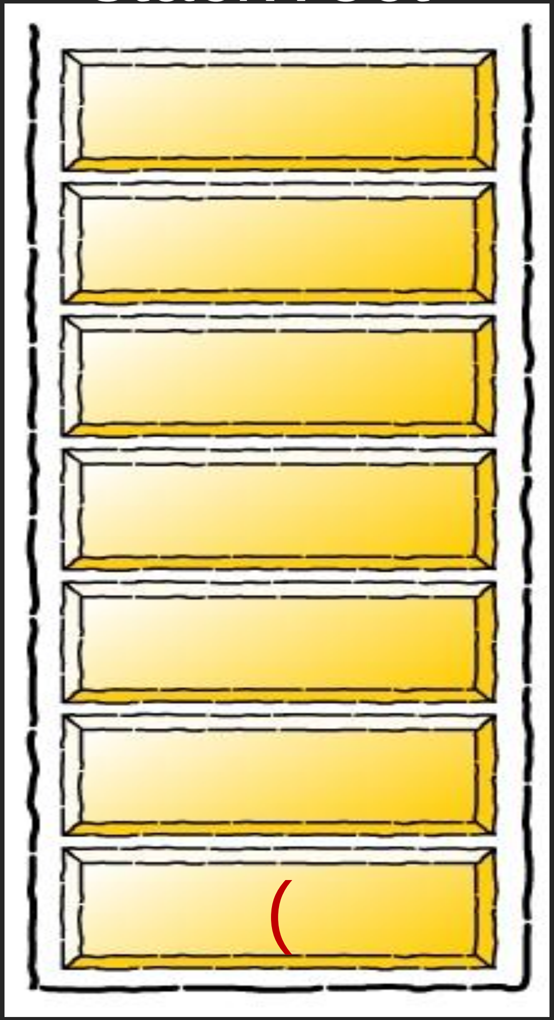


infixVect

a + b - c) * d – (e + f)

postfixVect

Infix to postfix conversion



infixVect

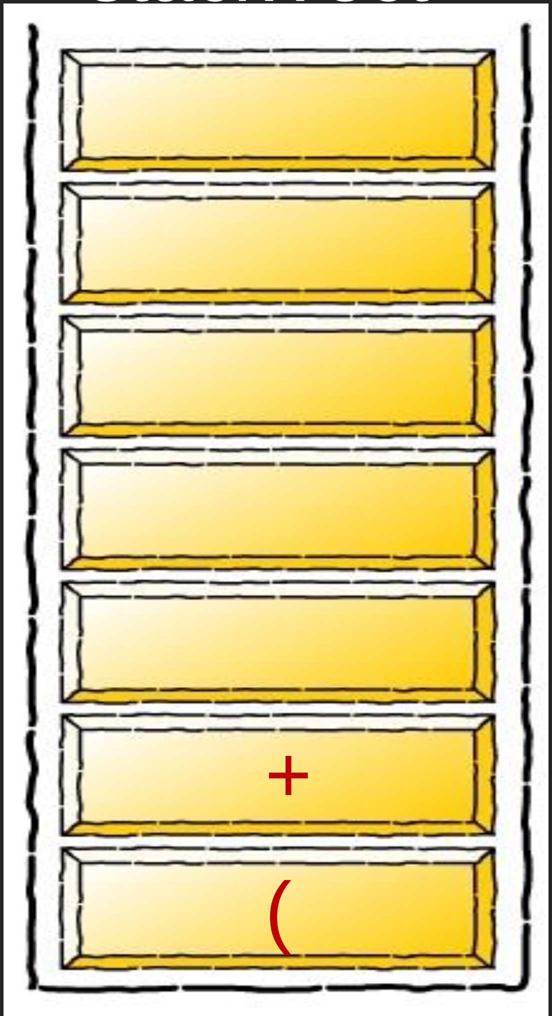
+ b - c) * d – (e + f)

postfixVect

a

Infix to postfix conversion

stackVect



+

(

infixVect

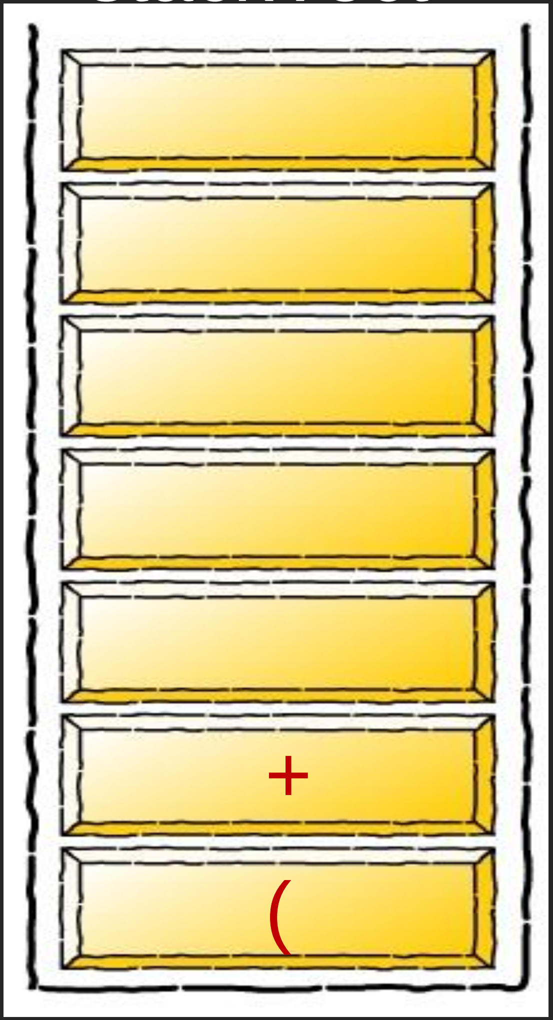
b - c) * d – (e + f)

postfixVect

a

Infix to postfix conversion

stackVect



+

(

infixVect

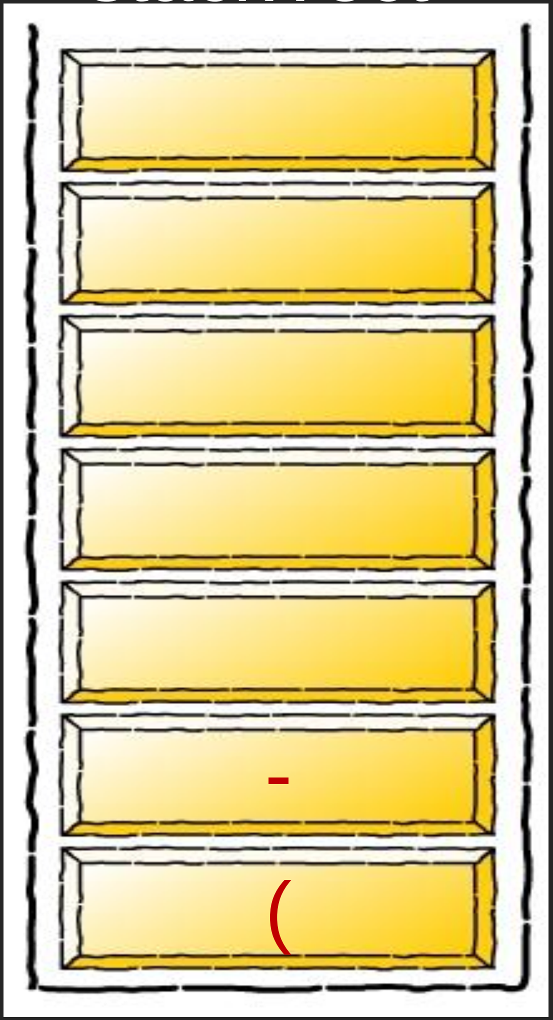
- c) * d – (e + f)

postfixVect

a b

Infix to postfix conversion

stackVect



infixVect

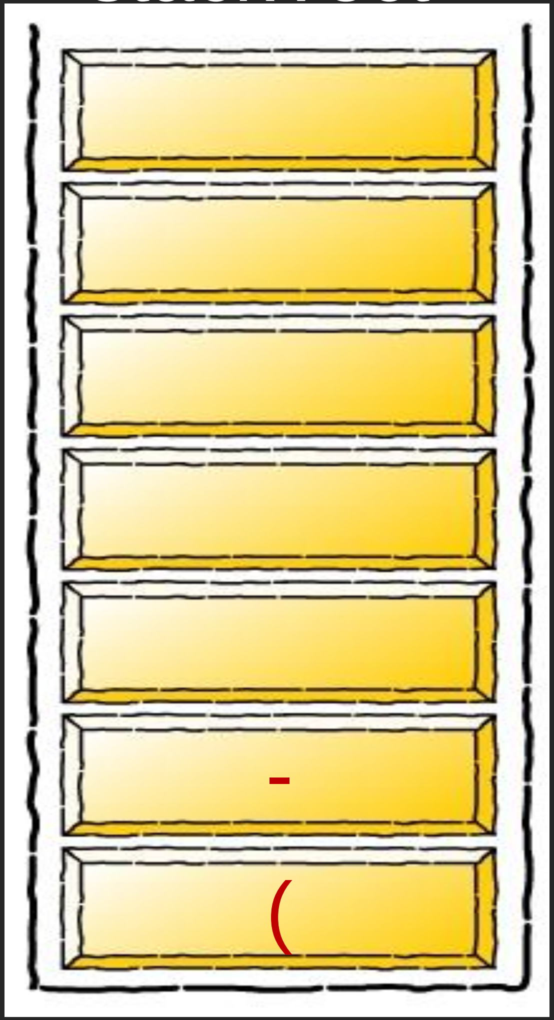
c) * d – (e + f)

postfixVect

a b +

Infix to postfix conversion

stackVect



-

(

infixVect

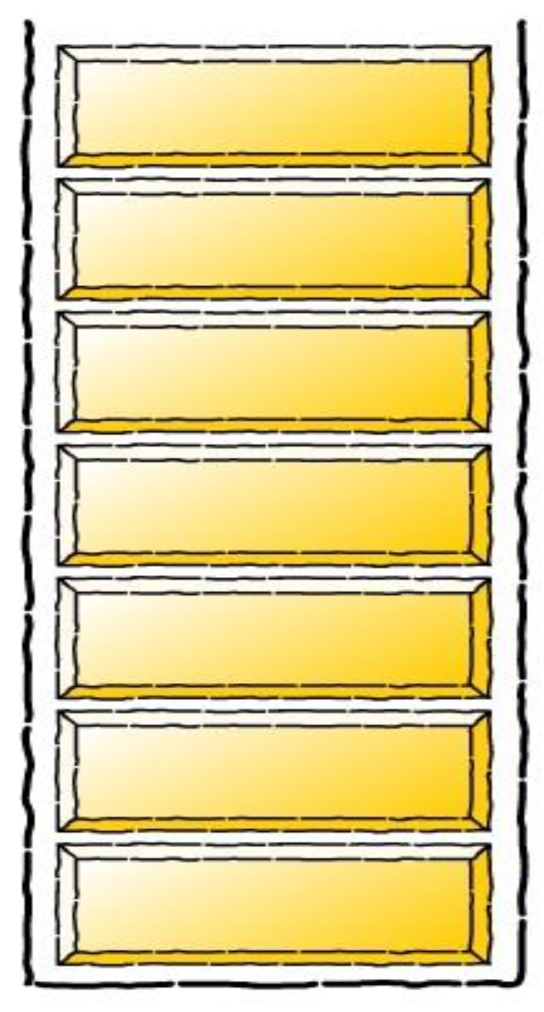
) * d – (e + f)

postfixVect

a b + c

Infix to postfix conversion

stackVect



infixVect

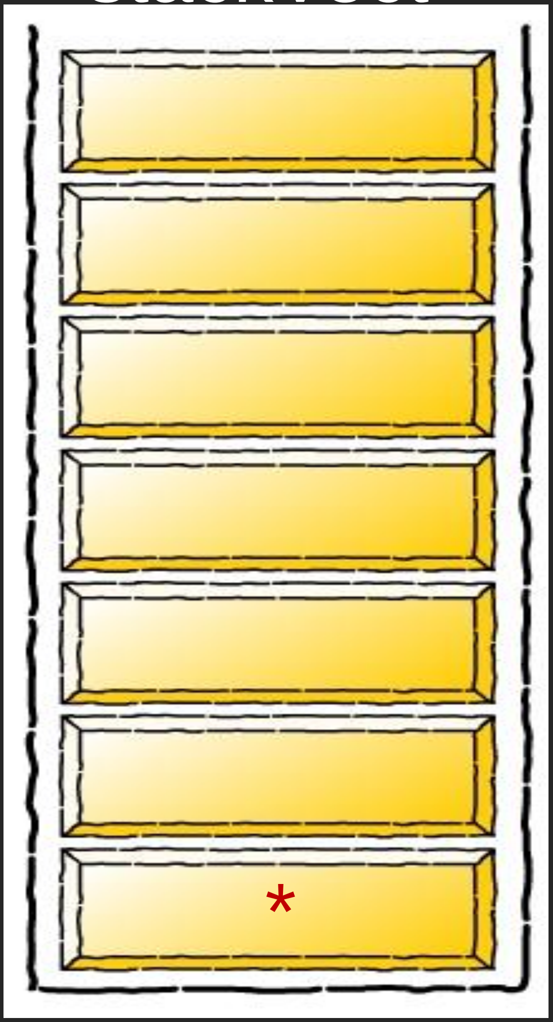
* d – (e + f)

postfixVect

a b + c -

Infix to postfix conversion

stackVect



infixVect

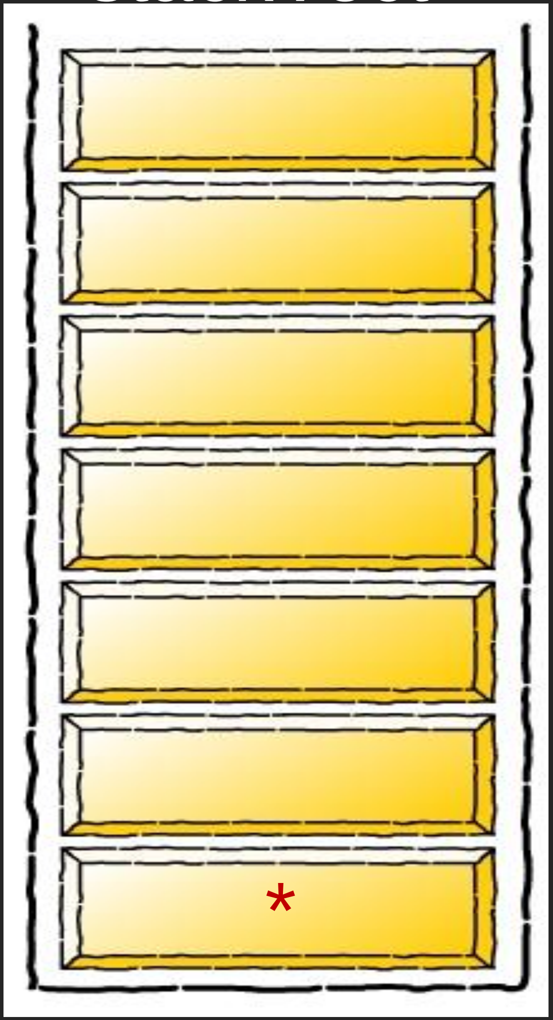
d – (e + f)

postfixVect

a b + c -

Infix to postfix conversion

stackVect



infixVect

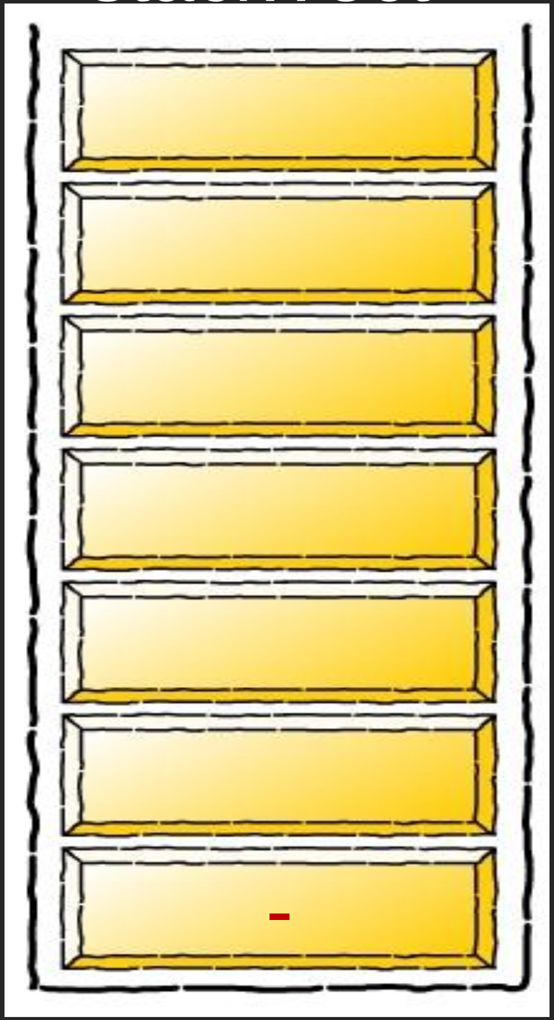
– (e + f)

postfixVect

a b + c - d

Infix to postfix conversion

stackVect



infixVect

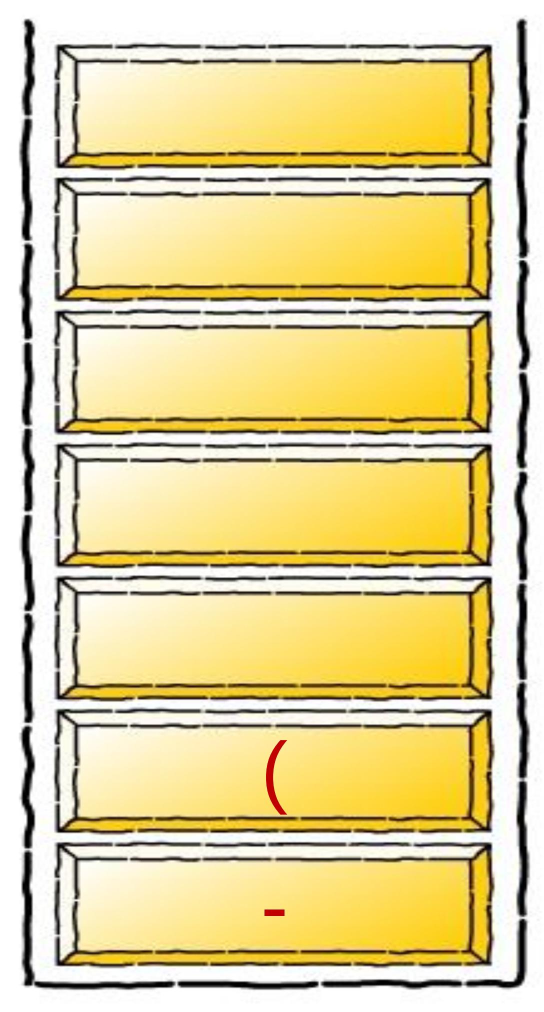
(e + f)

postfixVect

a b + c – d *

Infix to postfix conversion

stackVect



infixVect

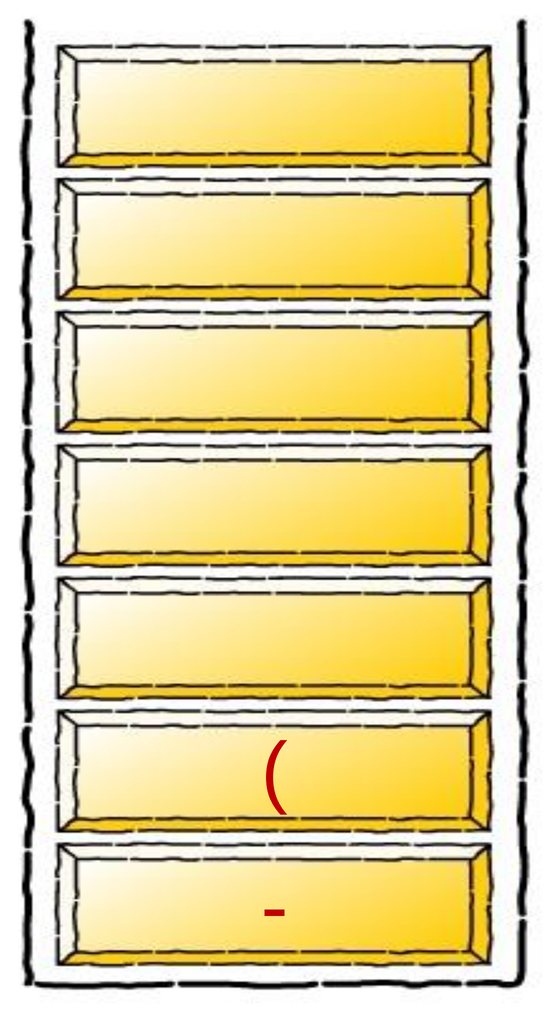
e + f)

postfixVect

a b + c – d *

Infix to postfix conversion

stackVect



(

-

infixVect

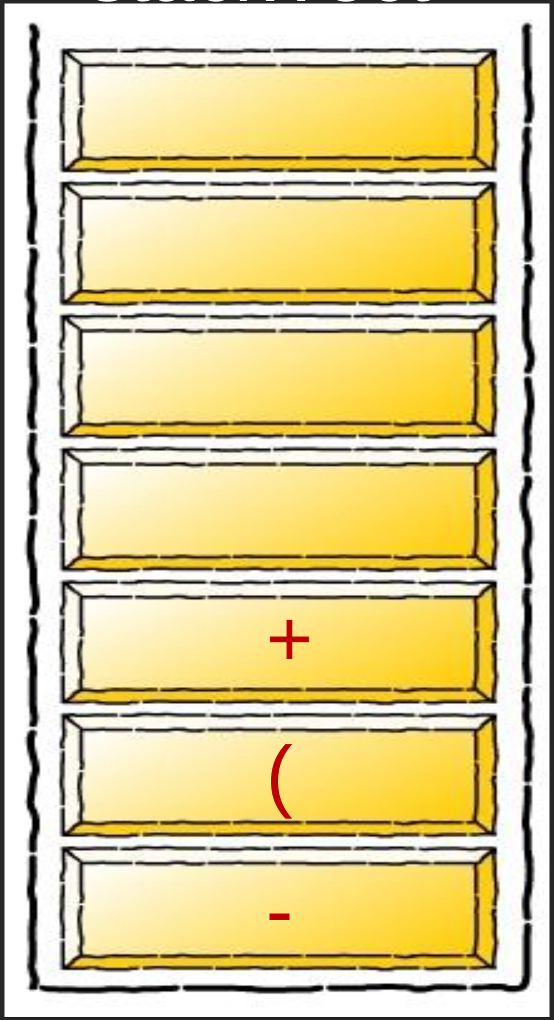
+ f)

postfixVect

a b + c – d * e

Infix to postfix conversion

stackVect



+

(

-

infixVect

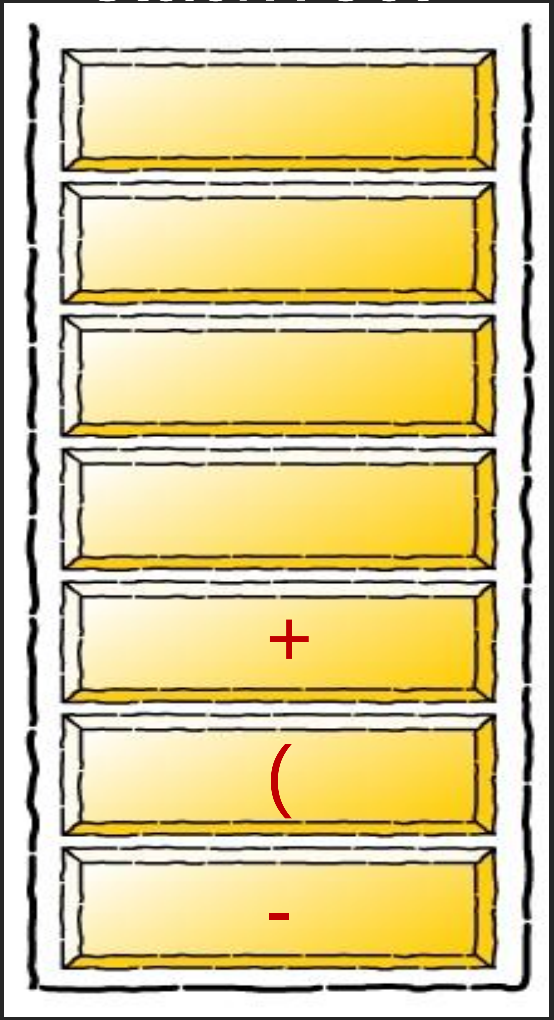
f)

postfixVect

a b + c – d * e

Infix to postfix conversion

stackVect



infixVect

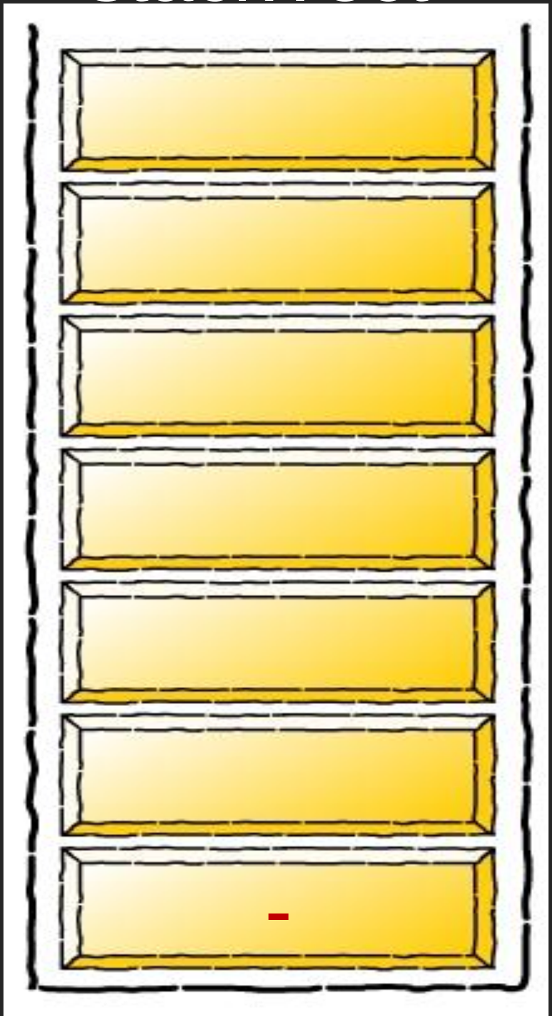
)

postfixVect

a b + c – d * e f

Infix to postfix conversion

stackVect



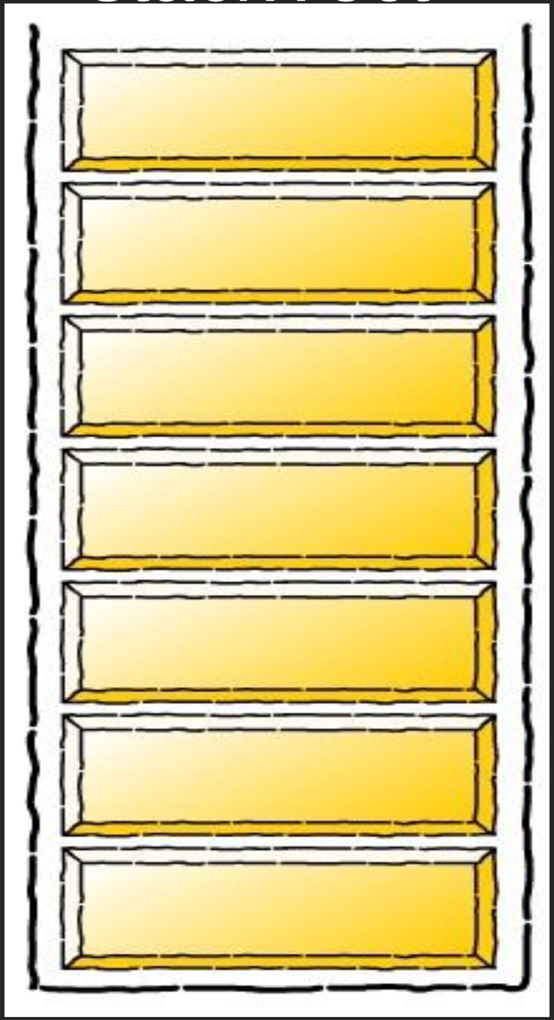
infixVect

postfixVect

a b + c – d * e f +

Infix to postfix conversion

stackVect



infixVect

postfixVect

a b + c – d * e f + -