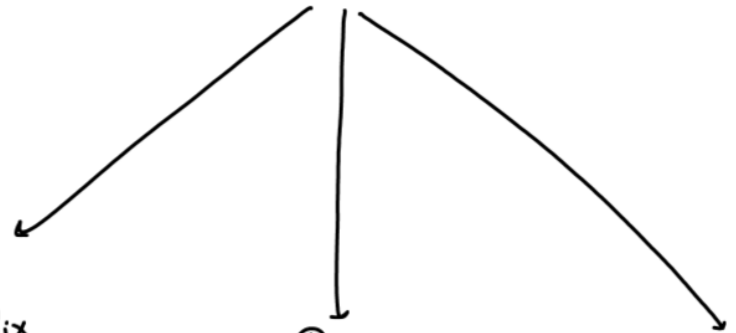
EXPRESSIONS



Infix	Postfix	Prefix
a + b	a b +	+ a b
a = b	a b =	= a b

Converting Infix To Postfix

Operater Precedence

( )

$

/ * %

+ -

Infix $A + \underline{B * C}$

$\underline{A + BC*}$

Postfix ABC*+

Infix $A + B * C / D - \underline{E \$ F}$

$A + \underline{B * C} / D - EF\$$

$A + \underline{BC* / D} - EF\$$

$\underline{A + BC*D/} - EF\$$

$ABC*D/+\ \underline{-\ EF\$}$

Postfix ABC*D/+EF$-

Infix To Prefix

Infix : A + B * C

A + *BC

Prefix : + A * B C

Infix A + B * C / D - E $ F

A + B * C / D - $EF

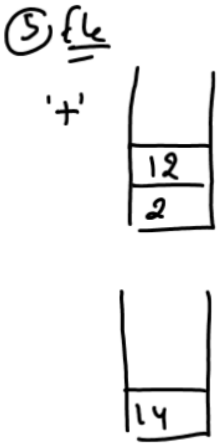
A + *BC / D - $EF

A + /*BCD - $EF

+A/*BCD - $EF

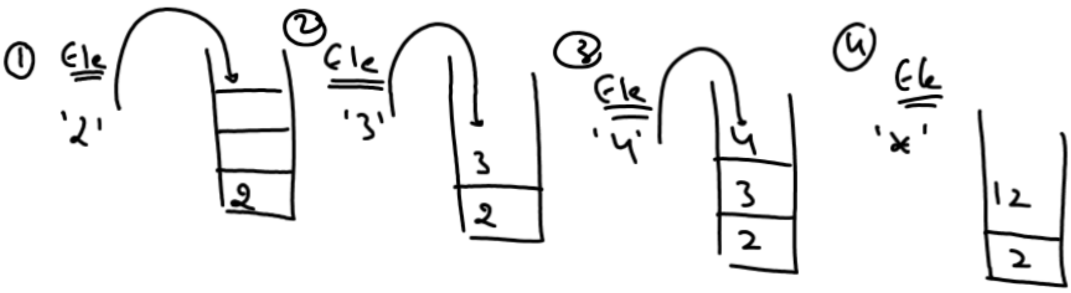
Prefix - + A / * B C D $ E F

Evaluating a Postfix Exp



2 + 3 * 4

0	1	2	3	4	5
'2'	'3'	'4'	'*'	'+'	'\0'



Infix   5 + 6 / 2 - 1 * 4 + 7

5 + 6 2 / - 1 * 4 + 7

5 + 6 2 / - 1 4 * + 7

5 6 2 / + - 1 4 * + 7

5 6 2 / + 1 4 * - + 7

Postfix   5 6 2 / + 1 4 * - 7 +

0	1	2	3	4	5	6	7	8	9	10	11
'5'	'6'	'2'	'/'	'+'	'1'	'4'	'x'	'-'	'7'	'+'	'\0'

Ele	Stack
'5'	5
'6'	5, 6
'2'	5, 6, 2
'/'	5, 3
'+'	8

Ele	Stack
'1'	8, 1
'4'	8, 1, 4
'x'	8, 4
'-'	4
'7'	4, 7
'+'	11 ← Ans
end	

Algorithm For Evaluating a Postfix Expression

1. Scan the given POSTFIX expression from LEFT to RIGHT one character at a time.

2. Check whether the character is an OPERAND or OPERATOR.

3. If it is an OPERAND , then PUSH it in the STACK

4. If it is an OPERATOR then:
    a. POP top 2 operands from the STACK
    b. Apply the operator on them
    c. PUSH the result back in the STACK

5. Repeat the above process until the POSTFIX expression finishes

6. POP the only remaining element from the STACK and return it as the answer of the expression
7. Finish