

Infix, Prefix & Postfix in Stack

1) \Rightarrow Infix $\rightarrow x+y$
 Postfix $\rightarrow xy+$
 Prefix $\rightarrow +xy$

Adv

- 1) Precedence & Associativity Rules not req
- 2) Can be evaluated in one traversal only.

2) $(x+y)*z \rightarrow *+xyz$ $xyz*+$
 $x+y*z \rightarrow +x*yz$ $xyz*+$

3) \rightarrow Infix to Postfix

a) Simple Sol

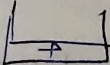
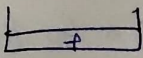
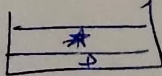
\Rightarrow Fully parenthesize the expression and then solve.

b) Efficient (Stack based)

\Rightarrow Input $\rightarrow a + b * c$

operator	Associativity
\wedge	Right to left
$*, /$	Left to right
$+, -$	"

↑
Precedence

<u>Input symbol(x)</u>	<u>Stack</u>	<u>Result (Postfix)</u>
a		a
+		a
b		ab
*		ab*
c	"	abc

Pop out everything \rightarrow answer is $abc*+$
one by one.

Algorithm

- ① Create an empty stack, st
- ② Do the following for every character x from left to right.
- ③ If x is:
 - a) An operand: Output it
 - b) Left parenthesis: Push to st
 - c) Right parenthesis: Pop from st until left parenthesis is found. Output the popped operators.
 - d) Operator: If stack is empty, push to st .
else (compare x with $st.top()$)
 - i) If x has higher precedence than $st.top$, push to st .
 - ii) If lower, pop $st.top$ and output until a lower precedence is found. Then push x to st .
 - iii) Equal precedence, use associativity and output everything from st .

Similarly for prefix.

- 1) Replace b) with c) & vice versa.
- 2) Go from right to left instead of left to right.
