Infix, Prefix & Postfix in Stack

1) ⇒ Infix → $x+y$
Postfix → $xy+$
Prefix → $+xy$

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

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

3) ⇒ **Infix to Postfix**

operator	Associativity
^	Right to left
*, /	left to right
+, −	"

↑ Precedence

a) Simple Sol

⇒ Fully paranthesize the expression and then solve.

b) Efficient (Stack based)

⇒ Input → a + b * c

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

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

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 paraenthesis : Pop from st until left Paraenthesis 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) $\Rightarrow$ Replace b) with c) & vice versa.

2) $\Rightarrow$ Go from right to left instead of left to right.