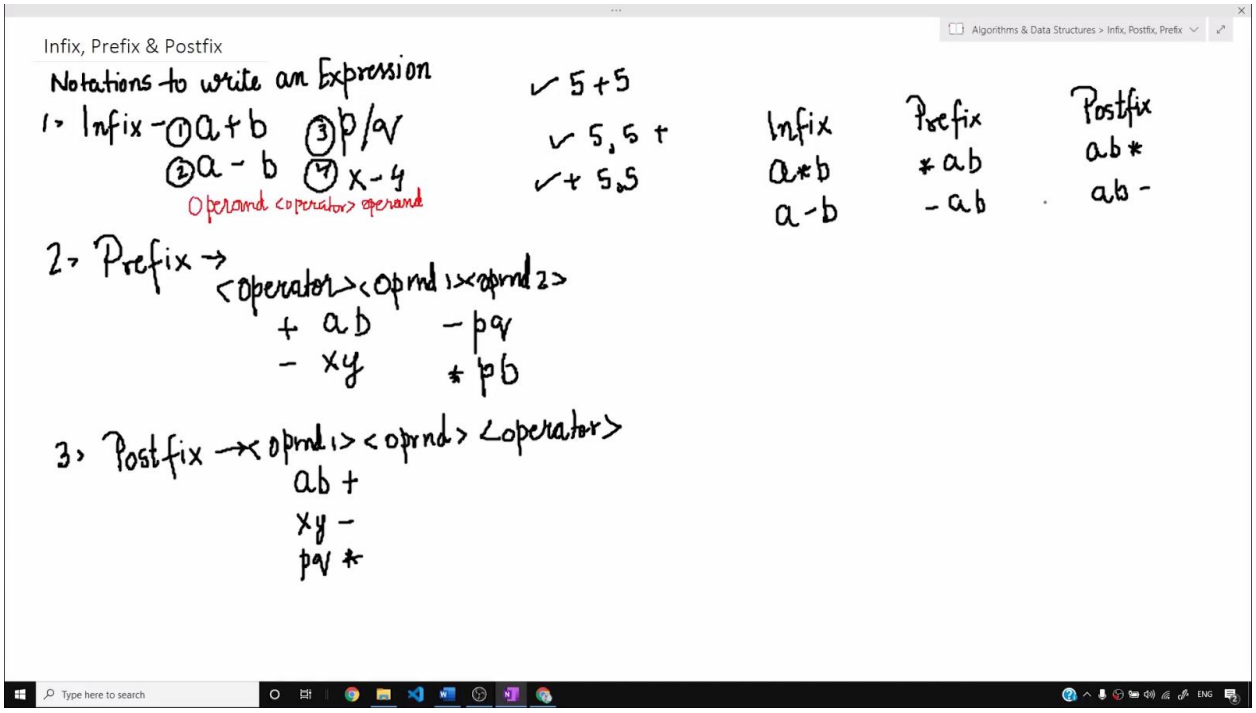

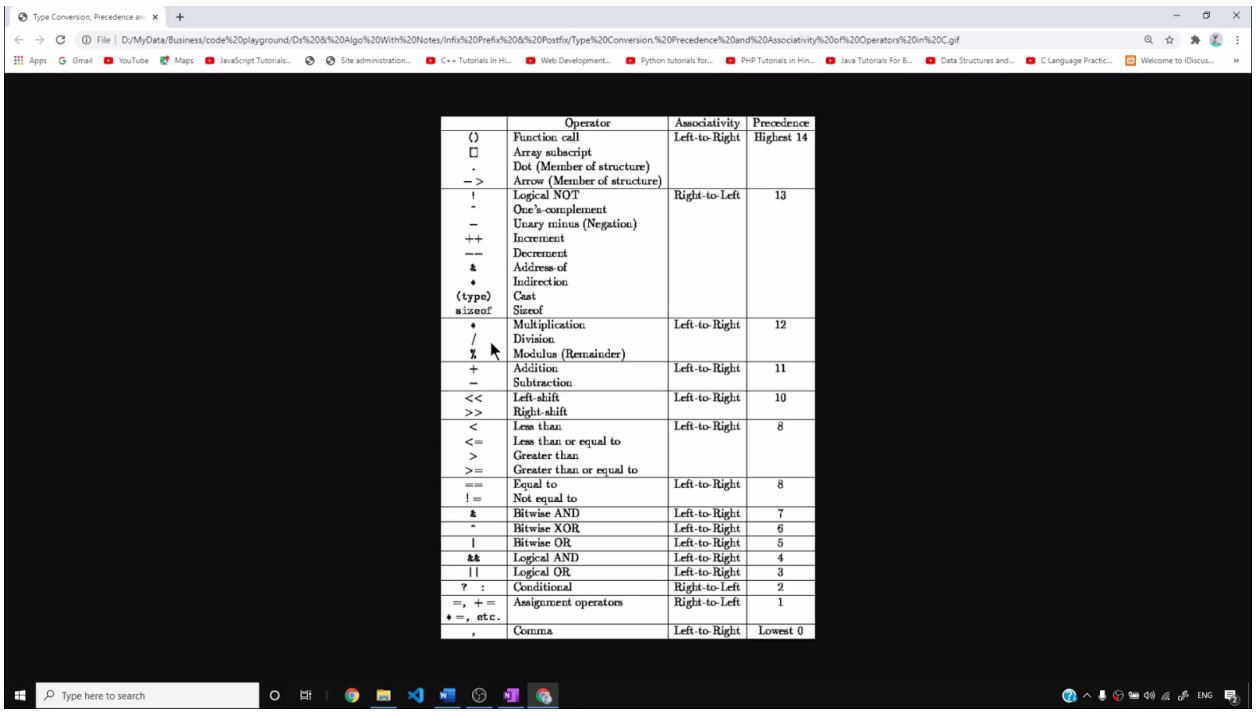

	Operator	Associativity	Precedence
()	Function call	Left-to-Right	Highest 14
[]	Array subscript		
.	Dot (Member of structure)		
->	Arrow (Member of structure)		
!	Logical NOT	Right-to-Left	13
~	One's-complement		
-	Unary minus (Negation)		
++	Increment		
--	Decrement		
&	Address-of		
*	Indirection		
(type)	Cast		
sizeof	Sizeof		
*	Multiplication	Left-to-Right	12
/	Division		
%	Modulus (Remainder)		
+	Addition	Left-to-Right	11
-	Subtraction		
<<	Left-shift	Left-to-Right	10
>>	Right-shift		
<	Less than	Left-to-Right	8
<=	Less than or equal to		
>	Greater than		
>=	Greater than or equal to		
==	Equal to	Left-to-Right	8
!=	Not equal to		
&	Bitwise AND	Left-to-Right	7
^	Bitwise XOR	Left-to-Right	6
|	Bitwise OR	Left-to-Right	5
&&	Logical AND	Left-to-Right	4
||	Logical OR	Left-to-Right	3
? :	Conditional	Right-to-Left	2
=, += *=, etc.	Assignment operators	Right-to-Left	1
,	Comma	Left-to-Right	Lowest 0

Infix, Prefix & Postfix

Left

Infix : (A * (B + C) * D) ✓

Postfix : ABC + * D *

A 8 * D *

7 D *

7 * D

operator	precedence
( )	3
* /	2
+ -	1

Right

Q1 x - y * z to prefix and postfix ?

① Prefix

S1 → Parenthesize the expression

(x - (y * z))

(x - [*yz])

- x * y z Done

② Postfix

S1 → Parenthesize the exp^n

(x - (y * z))

(x - [yz*])

xyz* -

③ p - q - r/a ✓

① Prefix ((p - q) - (r/a))

([- pq] - [/ra])

- - pq/ra = Prefix !

Quick Quiz : Convert this ③ to Postfix

Q $(m-n)*(p+q)$ to postfix ?

S1 → $((m-n)*(p+q))$

$([mn-]*[pq+])$

$[mn-][pq+]*$

$mn-pq+*$ → Postfix equivalent of this expression

Quick Quiz: Convert the same expression to prefix!