Infix, Prefix & Postfix:

→ Infix: This is the method we have all studying and applying for all our academic life. Here the operator comes in b/w two operands. And we say, two is added to three. For eg: 2+3, a*b, 6/3 etc.

<OPERAND 1><OPERATOR><OPERAND 2>

→ Prefix : This method might seem new to you, but we have vocally used them a lot as well. Here the operator comes before the two operands. And we say, Add two and three. For eg. +68, * x4, -32 etc.

<OPERATOR><OPERAND 1><OPERAND 2>

→ Postfix : This is the method that might as well seem new to you, but we have used even this in our communication. Here the operator comes after the two operands. And we say, Two & three are added. For e.g. 57+, ab*, 126/ etc.

<OPERAND 1><OPERAND 2><OPERATOR>

→ To understand the interchangeability of these terms, please refer to the table below:

	INFIX	PREFIX	POSTFIX
1)	a * b	*ab	ab*
2)	a - b	-ab	ab-

Page No.
Date

→ Why these methods?
→ COMPUTERS don't follow BODMAS; rather, they have their own operator precedence. And this is where we need these postfix and prefix notations. In programming, we use postfix notations more often, likewise, following the precedence order of machines.

1) x - y * z to prefix & postfix:

① Prefix

Step 1: Parenthesize the expression:

(x - (y * z))

(x - [* yz])

- x * yz

② Postfix

Step 1: Parenthesize the expression:

(x - (y * z))

(x - [yz *])

xyz * -

2) P - q - ↑ / a

① Prefix ((P - q) - (↑ / a))

([- Pq] - [/ ↑ a])

- - Pq / ↑ a

Page No. _____
Date | | | |

② Postfix

((P-q) - (m/a))

([Pq-) - [ma/])

Pq-ma/-

3) (m-n) * (P+q)

① Postfix :

⇒ ((m-n) * (P+q))

([mn-] * [Pq+])

[mn-][Pq+]*

mn-Pq+*

② PREFIX :

((m-n)* (P+q))

([-mn]* [+Pq])

*-mn+Pq