

Assignment No:- 03:-

Question No: 01

What is Infix Expression, Postfix expression and Prefix expression?

Infix Expression:-

When you write an arithmetic expression such as  $B * C$  So the form of expression in which operator appears b/w the operand. This type of expression is referred as infix expression.

for example:-

$$A + B * C$$

is an infix because operators  $+$  and  $*$  is b/w the operands



## Prefix:-

An expression in which all operators precedes the two operators they work on.

for Example:-  $+ A * BC$ .

## Post fix

An expression in which all the operators comes after the corresponding operands.

for Example:-

$$ABC * +$$



## Question No: 02:-

How the expressions are written and how each expression is solved?

Infix:  $A + B * C$

Prefix:  $+A \times B C$

Postfix:  $A B C * +$

There are two ways to solve this expression.



## QUESTION No: 03:-

How Infix is converted into postfix using Stack?

### RULES:-

- 1- first make 3 col:  
Infix      Stack      Postfix Exp.
- 2- If operand comes simply write it in postfix.
- 3- If an operation comes & stack is empty then simply push it.
- 4- "If both operations have same precedence" then check associativity.  
If associativity is left to right  
i) pop stack operation & check for next.  
ii) and push the incoming operation.  
If associativity is right to left.  
i) simply push the incoming oper.
- 5- If the operation has higher than stack operation simply push it.

live



6 If the incoming operation has lower precedence than stack operation i) pop out the stack and check again.

7 If the opening bracket "(" comes simply push it into the stack.

8 If opening bracket is on the top of the stack then whatever operation comes simply push it.

9) If the "Closing bracket" ")" comes, pop out the stack until you find an opening parenthesis, and (never write down opening and closing parenthesis in postfix.)

10 If the operand is finished then pop out the stack one by one until the stack is empty.



## Question No: 01:-

How Infix is converted to prefix using Stack? ("Reverse The String")

- 1- Make 3 column.
- 2- If operand come simply write it.
- 3- If operator come & stack is empty push it in stack.
- 4- If incoming operator has "higher" prec then simply push it in stack.
- 5- If incoming operator has "same" prec to stack operator  $\rightarrow$  LEFT  $\rightarrow$  RIGHT  
Push it into Stack.
- 6- If associativity is RIGHT  $\rightarrow$  LEFT.  
Pop out and check again - "<"  
If u find "Closing parenthesis" put into stack.
- 7- If closing parenthesis is on the top of the stack & any operator comes after this simply push it.



8- If you find "Opening Paranthesis" pop out the stack untill you find the Closing Paranthesis.  
Never write opening and Closing paranthesis.

9- If the incoming operator has lower precedence than stack operation.

Pop out the stack and check again untill you find same or lower

10 If expression is finished pop out all elements from the stack untill stack is empty.

11 "Reverse The Expression Again"



Q no 1:-

# Post Fix

$A + B * C \wedge (x - y * z) / (B + 2)$   
 $< Y * B - (C + A)^3$

Convert into Postfix expression using Stack.

Infix	Stack	Postfix
A	NULL	A
+	+	A
B	+ .	AB
*	+ *	AB
C	+ *	ABC
$\wedge$	+ * $\wedge$	ABC
(	+ * $\wedge$ (	ABCE
x	+ * $\wedge$ (	ABCLx
-	+ x $\wedge$ (-	ABCLx
y	+ x $\wedge$ (-	ABCLxy
*	+ x $\wedge$ (- *	ABCLxy
z	+ x $\wedge$ (- *	ABCLxyz
/	+ x $\wedge$ (- */	ABCLxyz*
(	+ x $\wedge$ (- */ (	ABCLxyz*
B	+ x $\wedge$ (- */ (	ABCLxyz*B
+	+ x $\wedge$ (- */ (+	ABCLxyz*B
2	+ x $\wedge$ (- */ (+	ABCLxyz*B2

)  
 )  
 <  
 y  
 \*  
 B  
 .  
 -  
 (



(B+2)  
B!=

)	+x^(-/	ABCxyz*B2+
)	+x^	ABCxyz*B2+/-
<	<	ABCxyz*B2+/-^*+
]	<	ABCxyz*B2+/-^*+]
*	<*	ABCxyz*B2+/-^*+]
B	<*	ABCxyz*B2+/-^*+
•		YB
-	<-	ABCxyz*B2+/-^*+
		YB*
(	<-(	ABCxyz*B2+/-^*+
		YB*
c	<-(	ABCxyz*B2+/-^
		*+YB*c
+	<-(+	" " "
D	<-(+	ABCxyz*B2+/-^
		*+YB*CD
)	<-	ABCxyz*B2+/-^*
		+YB*CD+
&&	&&	ABCxyz*B2+/-^
		*+YB*CD+ <del>A</del> -<
B	&&	ABCxyz*B2+/-^
		*+YB*CD+ <del>A</del> - <del>B</del> <
!=	&&!=	" " "
(	&&!=(	" " "
c	&&!=(	ABCxyz*B2+/-^
		*+YB*CD+ <del>A</del> - <del>B</del> <



+	$xyz = ( +$	$xyz$	$xyz$
A	$xyz = ( +$	$ABCxyz * B2 + / - \wedge$	$ABCxyz * B2 + / - \wedge$
)	$xyz =$	$* + yB * CD + \bar{0} < CA$	$* + yB * CD + \bar{0} < CA$
A	$xyz = \wedge$	$ABCxyz * B2 + / - \wedge$	$ABCxyz * B2 + / - \wedge$
		$* + yB * CD + \bar{0} < CA$	$* + yB * CD + \bar{0} < CA$
		$+ \bullet$	$+ \bullet$
3	$xyz = \wedge$	$ABCxyz * B2 + / - \wedge$	$ABCxyz * B2 + / - \wedge$
		$* \bullet + yB * CD + \bar{0} <$	$* \bullet + yB * CD + \bar{0} <$
		$BCA + \bullet 3^{\wedge} ! = 8 \&$	$BCA + \bullet 3^{\wedge} ! = 8 \&$

Post fix :-

$$ABCxyz * B2 + / - \wedge * + yB * CD + \bar{0} < \\ \bullet BCA + \bullet 3^{\wedge} ! = 8 \&$$



# Prefix:-

CONVERTING Infix to Prefix Using Stack:-

- 1-  $A + B * C \wedge (X - Y * Z / (B + 2)) < Y * B$   
 $- (C + D) \& \& B != (C + A) \wedge 3.$

Infix Stack Prefix.

Reverse:-  $3 \wedge) A + C ( ! = B \& \& ) D$   
 $+ C (- B * Y <) ) 2 + B ( / Z * Y -$   
 $X ( \wedge C * B + A$

Infix	Stack	Prefix exp.
3	NULL	3
$\wedge$	$\wedge$	"
)	$\wedge)$	"
A	$\wedge)$	3A
+	$\wedge)+$	3A
C	$\wedge)+$	3AC
(	$\wedge$	3AC+
$!=$	$!=$	3AC+^
B	$!=$	3AC+^B
$\&\&$	$\&\&$	3AC+^B!=
)	$\&\&)$	" "
D	$\&\&)$	3AC+^B!=D
+	$\&\&)+$	" "
C	"	3AC+^B!=DC







Infix	Stack	Prefix
x	"	$3AC + \wedge B! = DC + BY$ $* - 2B + ZY * / x$
(	&&<	$3AC + \wedge B! = DC + BY$ $* - 2B + ZY * / x -$
^	&&<^	"
C	"	$3AC + \wedge B! = DC + BY$ $* - 2B + ZY * / x - C$
*	&&<*	$3AC + \wedge B! = DC + BY$ $* - 2B + ZY * / x - C$ ^
B	" "	$3AC + \wedge B! = DC + BY$ $* - 2B + ZY * / x - C$ ^B
+	&&<+	$3AC + \wedge B! = DC + BY$ $* - 2B + ZY * / x - C$ ^B*
A	&&'<+	$3AC + \wedge B! = DC +$ $BY * - 2B + ZY * / x - C$ &&.

~~Prefix~~ Reverse Now

$$3AC + \wedge B! = DC + BY$$

$$* - 2B + ZY * / x - C \wedge B * A$$

$$+ < &&$$

$$! = DC +$$

"

$$! = DC + B$$

"

$$B! = DC + B$$

$$B! = DC + B$$

"

"

$$B! = DC + B$$

$$B! = DC + B$$

$$B! = DC + B$$

+

"

$$\wedge B! = DC +$$

+ Z.

"

$$\wedge B! = DC +$$

$$B + ZY$$

$$\wedge B! = DC +$$

$$B + ZY *$$



The final Prefix Eq, would be,

$$E \& \leq + A * B \wedge C - x / * yz + B2$$

$$- * yB + CD! = B \wedge + C A3$$



# Solution Of Postfix

EVALUATION OF POSTFIX USING STACK :-

ABCX YZ \* B2 + / - ^ \* + YB  
\* CD + - < BCA + 3 ^ ! = 8 8

Putting The Values :-

A=1, B=2, Y=2, Z=4, X=5  
C=1, D=2

1 2 1 5 2 4 \* 2 2 + / - ^ \* + 2 2  
\* 1 2 + - < 2 1 1 + 3 ^ ! = 8 8

Input	Action perform	Stack
-------	----------------	-------

1	Push 1	4
2	Push 2	2
1	Push 1	5
5	Push 5	1
2	Push 2	2
4	Push 4	1

*	Pop (4, 2)	8
	Push (8)	5
		1
		2
		1



Stack = 0 initially

1  
2

Push 2  
Push 2

+

Pop (2, 2)  
Push 4

/

Pop (4, 8)  
Push (2)

-

Pop (2, 5)  
Push (3)

<

Pop (3, 1)  
Push (1)

2
2
5
1
2
4

2
5
1
2
1

3
1
2
1

1
2
1

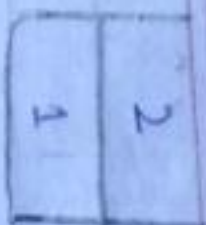
\* + 2 2 \* 1 2 + - <



\*

Pop (1, 2)

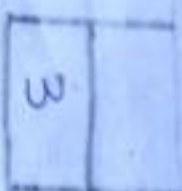
Push (2)



+

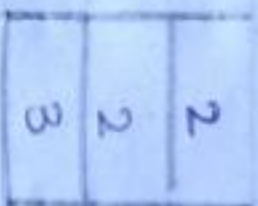
Pop (2, 1)

Push (3)



2

Push 2



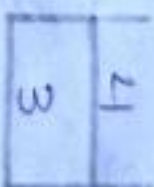
2

Push 2

\*

Pop (2, 2)

Push (4)



Push (1)

Push (2)



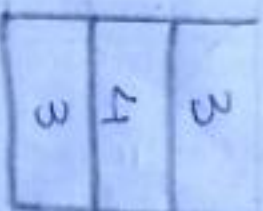
1

2

+

Pop (2, 1)

Push (3)



-

Pop (3, 4)

Push (1)

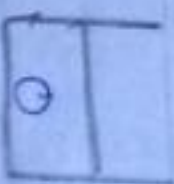


<

Pop (1, 3)

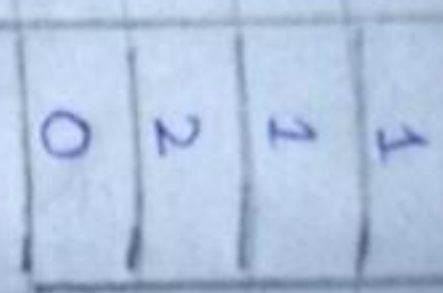
Check (3 < 1)

false = 0



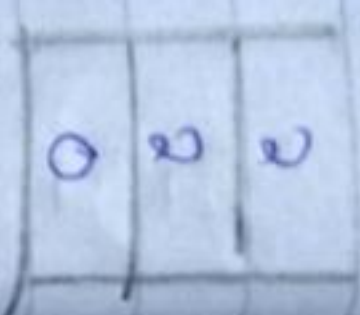


2  
1  
1  
Push(2)  
Push(1)  
Push(1)

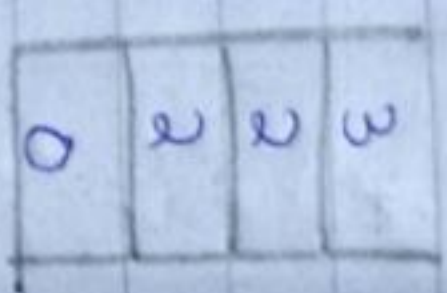


+

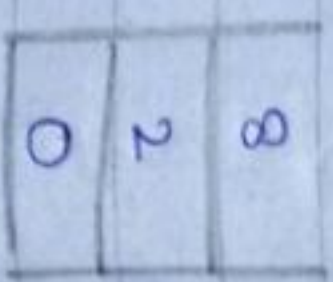
Pop(1,1)  
Push(2)



3  
Push 3



^  
Pop(3,2)  
Push(8)



j =  
Pop(8,2)

Check (2,1=8)  
True = 1



&&  
Pop(1,0)

Check (0 & 1)  
false = 0



Postfix = 0 Ans.



# Post Fix

Q NO: 2

$$2 - A < (B - C) * D \wedge (2 + x) \parallel A != x \&\& \\ C == x * (C - 2)$$

Infix	Stack	Postfix.
A	NULL	A
<	<	A
(	< (	A
B	< (	AB
-	< (-	AB
C	< (-	ABC
)	<	ABC-
*	< *	ABC-
D	< *	ABC-D
$\wedge$	< * $\wedge$	ABC-D
(	< * $\wedge$ (	ABC-D
2	< * $\wedge$ (	ABC-D2
+	< * $\wedge$ (+	ABC-D2
x	< * $\wedge$ (+	ABC-D2x
)	< * $\wedge$ .	ABC-D2x+
$\parallel$	$\parallel$	ABC-D2x+ $\wedge$ *<
A	$\parallel$	ABC-D2x+ $\wedge$ *<A
!=	$\parallel$ !=	" " " "
x	$\parallel$ !=	ABC-D2x+ $\wedge$ *<Ax
&&	$\parallel$ &&	ABC-D2x+ $\wedge$ *<Ax
		!=







# Prefix Q No: 2

CONVERTING TO "PREFIX" USING  
STACK :-

2-  $A < (B - C) * D^{(2+x)} \parallel A != x \&\&$   
 $C == x * (C - 2)$

Reverse the Strings:-

$) 2 - C ( * x == C \&\& x !=$   
 $A \parallel ) x + 2 ( A D * ) C -$   
 $B ( < A.$

Infix	Stack	Prefix.
)	)	NULL
2	)	2
-	) -	2
C	) -	2 C
(	NULL	2 C -
*	*	2 C -
x	*	2 C - x
==	==	2 C - x *
C	==	2 C - x * C
&&	&&	2 C - x * C ==
x	&&	2 C - x * C == x
!=	&& !=	2 C - x * C == x
A	&& !=	2 C - x * C == x A



S: 0VT 10 x173710

S: 024

Infix	Stack	Prefix
		2C-x* C==xA != &&
)	)	2C-x * C == xA != &&
x	)	2C-x * C == xA != && x
+	) +	2C-x * C == xA != && x
2	) +	2C-x * C == xA != && x 2
(		2C-x * C == xA != && x 2 +
A	^	4 4
D	^	2C-x * C == != && x 2 +
*	*	2C-x * C == != && x 2 +

Infix
)
C
-
B
(
<
A
final
<del>AX</del>
AX



# Case 8

Infix	Stack	Prefix
)	* )	2C - x * C == xA != && x2 + D^
C	* )	2C - x * C == xA != && x2 + D^ C
-	* ) -	2C - x * C == xA != && x2 + D^ C
B	* ) -	2C - x * C == xA != && x2 + D^ C B
(	*	2C - x * C == xA != && x2 + D^ C B -
<	<	2C - x * C == xA != && x2 + D^ C B - *
A	<	2C - x * C == xA != && x2 + D^ C B - * A <

final Prefix Expression after Rev.

~~U < A \* - B C ^ D + 2 x && !=~~

$$A x == C * x - C 2.$$



# Solution Of Prefix Q No 2

## EVALUATION OF PREFIX EXPRESSION

→ USING STACK →

$$!! < A * - BC^D + 2x \&\& != Ax$$

$$== C * x - C2.$$

Substituting  $A=1, B=2, Y=2, Z=4$

$$x=5, C=1, D=2.$$

$$!! < 1 * - 21^2 + 25 \&\& != 15$$

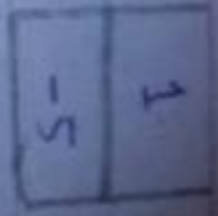
$$== 1 * 5 - 12.$$

Input	Action performed	Stack
2	Push	1
1	Push	2
-	Pop (1, 2) Push (-1)	-1
5	Push	5 -1
*	Pop (5, -1) Push (-5)	-5



1

Push



==

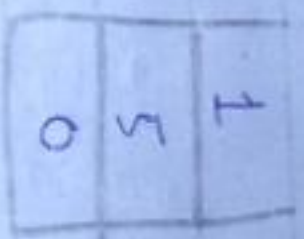
Pop (1, -5)  
(1 == -5)



false = 0

5

Push



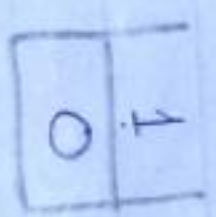
1

Push

i =

Pop (1, 5)

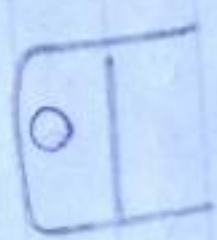
Check (1, 1 == 5)  
true = 1



2, 2

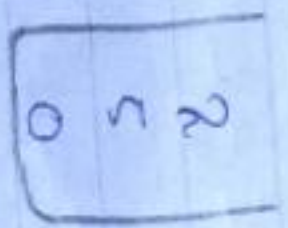
Pop (1, 0)

Check (1, 2 & 2)



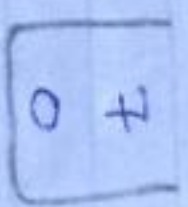
false = 0

Push

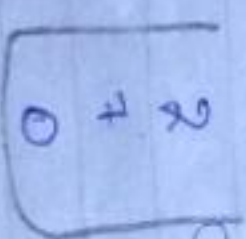


Push

Pop (2, 5)  
Push (7)



Push

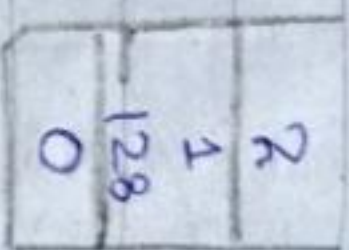




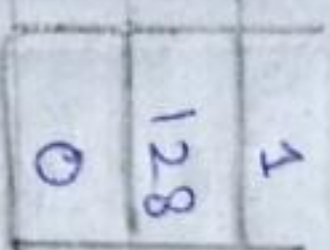
Pop (2, 7)  
Push (2, 7)



Push



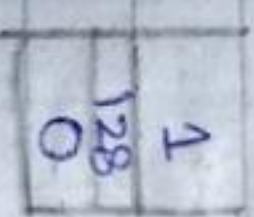
Pop (2, 1)  
Push (1)



Pop (1, 128)  
Push (128)



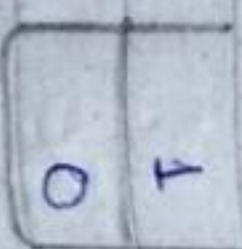
Push



Pop (1, 128)

Check (1 < 128)

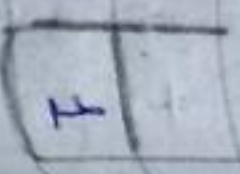
if true = 1



Pop (1, 0)

Check (1 == 0)

True = 1



✓

= 1

AD