1) Infix Expression + - operators <operand > operator <operands Ex: A+B Postfix Expression
 A
 B
 Coperands
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
 C
<operand 17 < operand 27 < operators A+B Ex: AB+ 3 Prefix Expression <operator><operandi><operandi> EN TAB

Infix to Postfix Conversion

1. Evaluating Arithmetic Expression

To evaluate an arithmetic expressions, first convert the given infix expression to postfix expression and then evaluate the postfix expression using stack.

Infix to Postfix Conversion

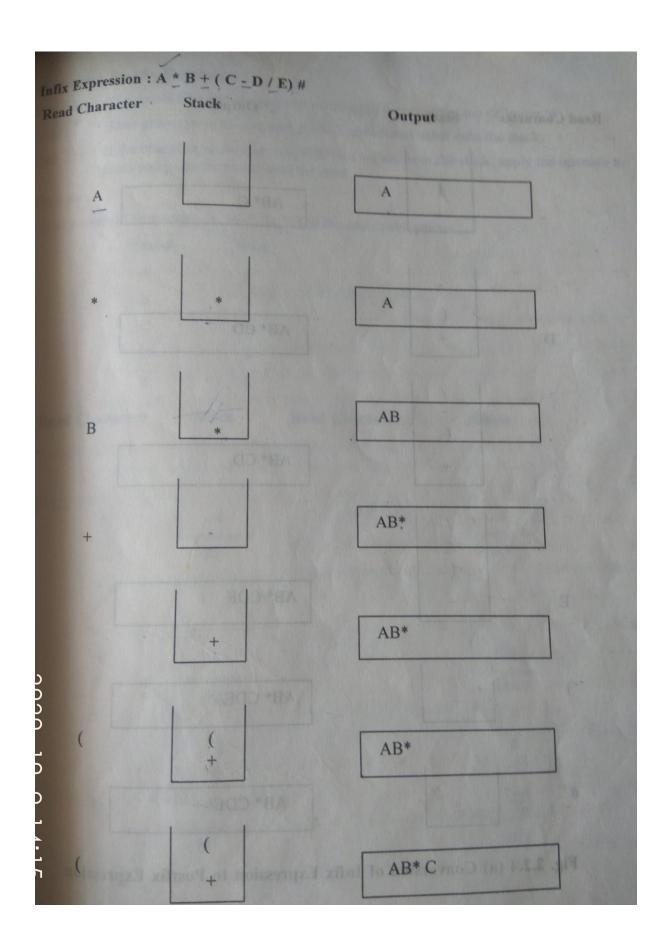
Read the infix expression one character at a time until it encounters the delimiter. "#"

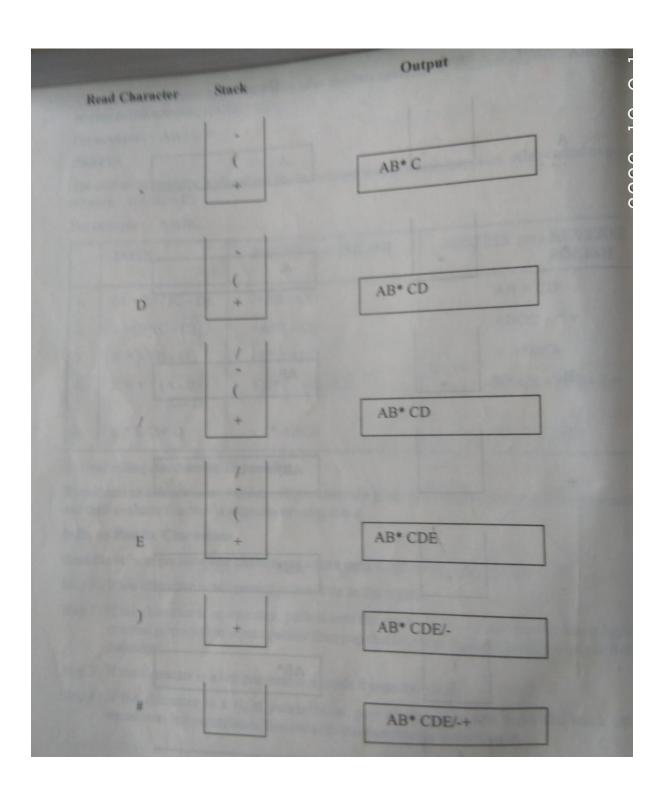
Step 1 : If the character is an operand, place it on to the output.

Step 2 : If the character is an operator, push it onto the stack. If the stack operator has a higher or equal priority than input operator then pop that operator from the stack and place it onto the output.

Step 3 : If the character is a left paraenthesis, push it onto the stack.

Step 4: If the character is a right paraenthesis, pop all the operators from the stack till it encounters left parenthesis, discard both the parenthesis in the output.





A+ B/C+(D+E	J-F)		priority.
ymbol.	Sterk	Post fin .	$ \begin{array}{c} \wedge \rightarrow 3 \\ f_{1} \end{pmatrix} 2 $
6	L		+,- → 1
A +	(+	A	- No two operator of
B/	(+/	A B	Same priority can stay together in the Stack calumn.
C	(+1	ABC	Stack column.
c * ((+#	ABC/	-
D +	(++) (++) (++) (+)	ABC/D.	
F	(+*(+	ABC /DE	
)	(+* (+)	ABC/DE+	
F	(4-	ABC/DE+#+	
>	(-	ABC/DE+++	
-	5	ABC / DE +++ F	

Evaluating Postfix Expression

Read the postfix Expression one character at a time until encounters the delimiter "#"

Step 1:If the character is an operand , push its associated value onto the stack

Step 2: If the character is an operator, pop two values from the stack , apply the operator to them and push the result onto the stack

AB*CDE/-+

	Value				
A	4				
B	A HOLDS HA				
С	5				
D	8				
E	2				
Read Character	Stack	Read Character	ang symbo	hack	
a symbol in the starts, t	nd ng openin		ing symbo	from the d	
ed sy mbola.	as nusmannes		and all all all all all all all all all al	5	
the second s		er stack is not empty, re thance gymbols.	citori as B	4	
A	4				
	(i)			(11)	
	Stack		of Emply	In the second	
	100		1	5	
The Part	1.2.4 (8) 10	C C	steph Sype	20	
*	20		L		
	(iii)		(i)	(iv)	
				2	
	8		100	8	
	5			5	
D	20	E		20	
	(v)			(vi)	

1	ATTAC L'AND
	inating Postfix Expression
	Libe posifix expression and character at a
hits associated val. 5 only the stock.	alt - If the character is an operand, pus
20	20 11 the bargeter an operator, FO
(vii)	(viii)
the associated alless in a	au consider the symbols A. B. C. D. F. had
+ 21	Result = 21°
(xi) Fig. 2.2.4 (b) Evaluat	tion of AB*CDE / - +

stalk

2/3=8 8/8=1 2 * 3= 6

The following postfix expression with single digit operands is evaluated using a stack:

823 ^/ 23*+51*-Note that ^ is the exponentiation operator. The top two elements of the stack after the first * is evaluated are

a) 6,1	b) 5,7
c) 3,2	d) 1,5