EXPRESSION TREES

The expression tree for the expression (a+b*c) + ((d*e+f) *g) will be shown in following figure.

The leaves of an expression tree are operands such as constants or variables and the Expression tree is a binary tree since all the operations are binary operations.

Here

(a + b * c) + ((d * e + f) * g)

Left subtree

Right subtree

Root.

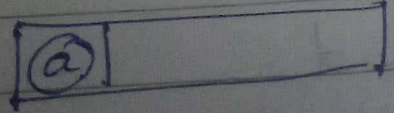
Rules to Construct a Postfix expression.

1> Read one symbol at a time

2) Check whether the symbol is operator or operand.

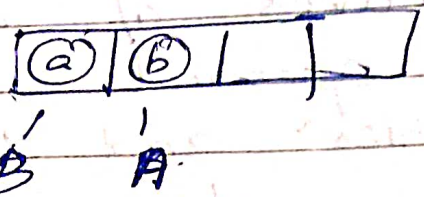
a) If the symbol is operand, push it to the stack.
(Push the operand as node)

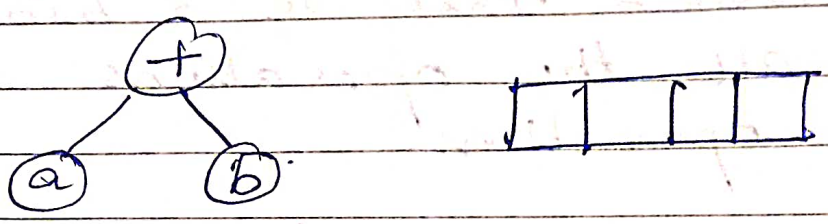
Eg:-



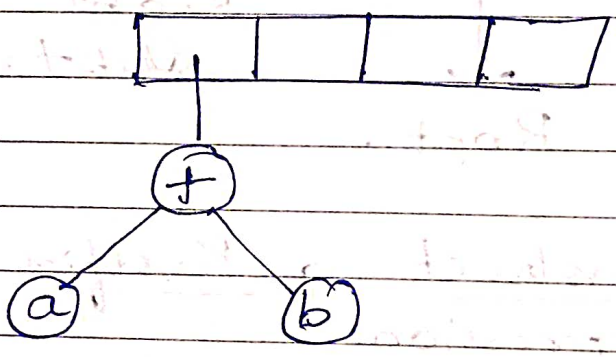
(b) if the symbol is operand ~~to~~
pop top two operands from the stack. (A & B)
create a tree with operator as root & B as left child & A as right child.

Eg:-





The push the resulting tree onto the stack.



Example.

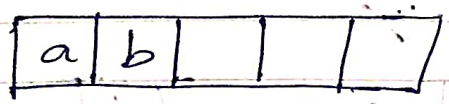
The postfix expression ~~for the given infix expression~~ is

ab+ cde + ** .

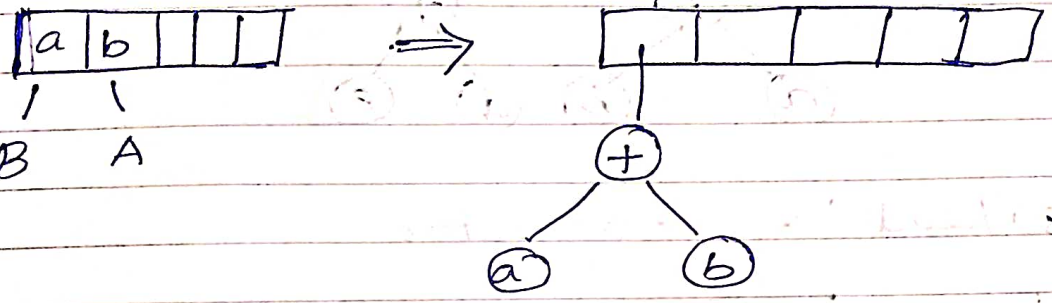
1) Read 'a'
'a' is operand.
Push it to stack.

(Note: The size of stack is equal to the number of operands)

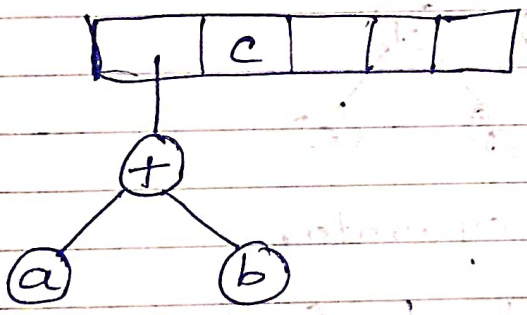
2) Read 'b' → Operand
Push it to stack.



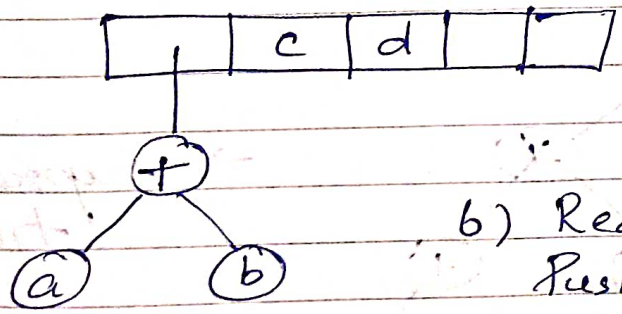
3) Read '+' → operator.



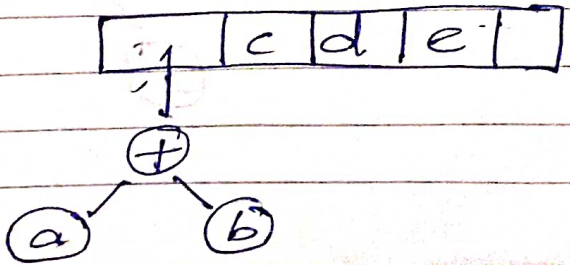
4) Read 'c' → operand
Push it to the stack.



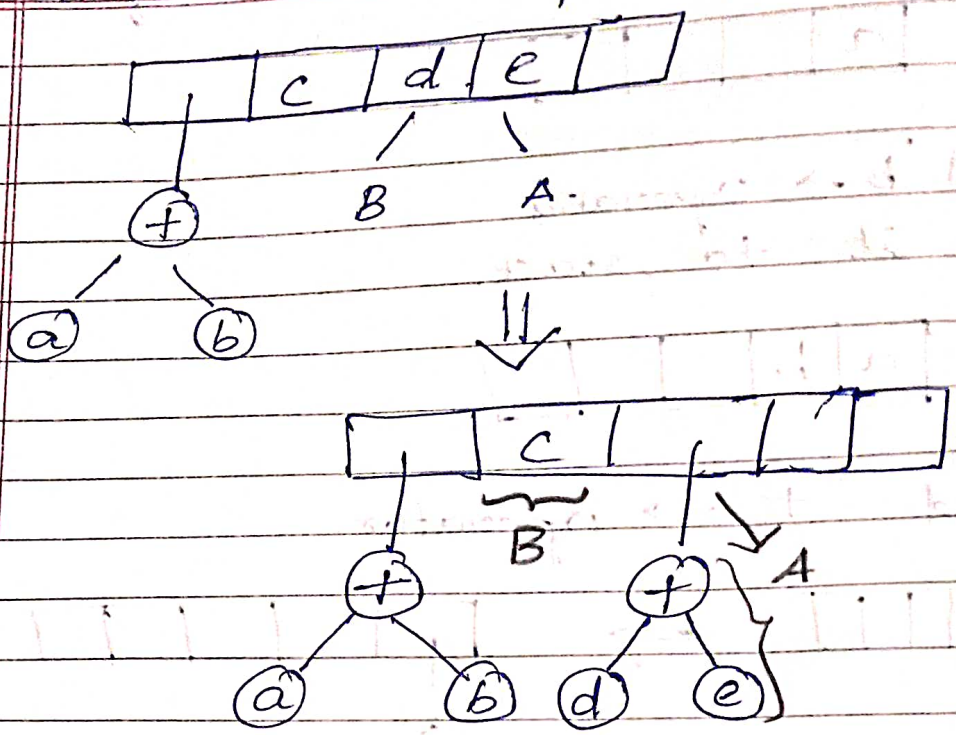
5) Read 'd' → operand. : Push it to stack.



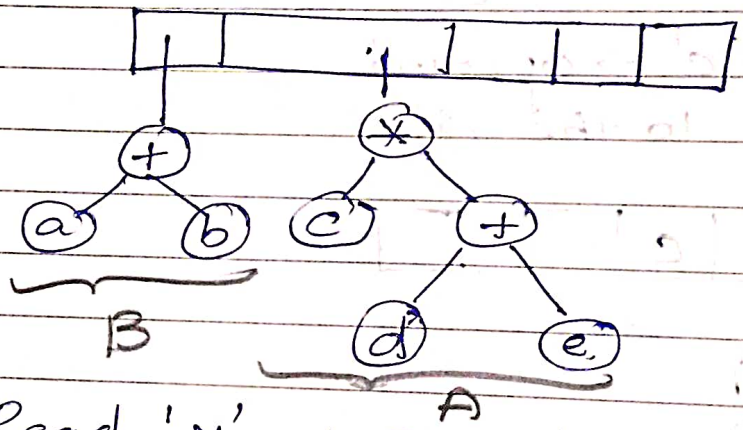
6) Read 'e' → operand.
Push it to stack.



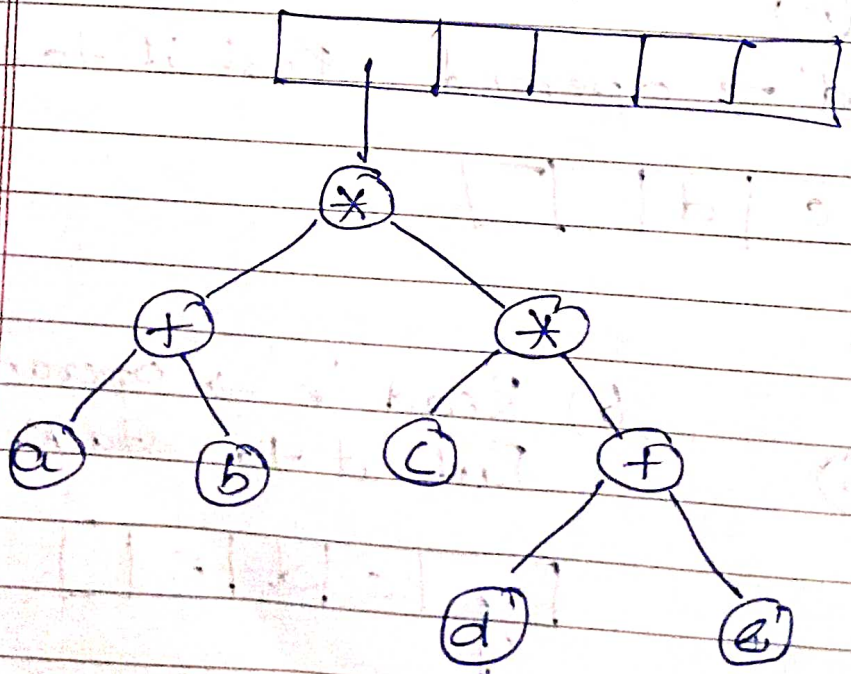
7) Read '+' → operator



8) Read '*' → operator



9) Read '+' → operator



→ Expression tree