*Stack can be used to reverse any input.

Applications of Stack

① Infix ~~App~~ expression →

operands → A, B, --   operator → + - * ~~⊘~~ /

<operand> <operator> <operand> → A + B

② P~~re~~ostfix expression   ③ Prefix exp →

(A+B)   AB+   +AB   (A+B)

conv. of Infix to Postfix exp (Algorithm) :-

STEPS →

① If the char is left paranthesis, push to the stack.
② If the char is OPERAND, add to the POSTFIX exp.
③ If the char is OPERATOR, check whether stack is empty.
if empty - push to stack. If not, check priority
(i) If priority of OPERATOR > OPERATOR present at TOP of stack, then PUSH the operator in the stack.
(ii) If the priority of operator ≤ operator present at TOP of STACK, then pop the operator from stack and add to POSTFIX exp and goto step (i)

If its left para. don't check push push in stack

④ If the char is RIGHT Paranthesis, then pop all the operators from the stack until it reaches left par & add to postfix.
⑤ After reading all char, if stack is not empty then pop and add to postfix.

for eg A+B*C

A → char (operand) by Step 2

+ → operator Step3

→ stack is empty - push to stack

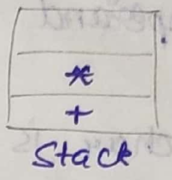
B → operand Step 2

* → operator Step 3 (i)

C → operand Step 2

Step 5 $\boxed{ABC*+}$

Postfix $\boxed{A\ B\ C}$



eg2 a-(b*c-d)/e → Infix

char	stack	postfix
a		a
-	-	a
(	(-	a
b	(-	ab
*	* (-	ab
c	* (-	abc
-	- (-	abc*
d	- (-	abc*d
)	-	abc*d-
/	/ -	abc*d-
e	/ -	abc*d-e
		abc*d-e/-

(→ push

A → postfix

+ → empty - Push

not empty

is of ↑ priority push

↓ pop before & postfix it check again

) → pop all till (add to postfix

last → pop, add to postfix

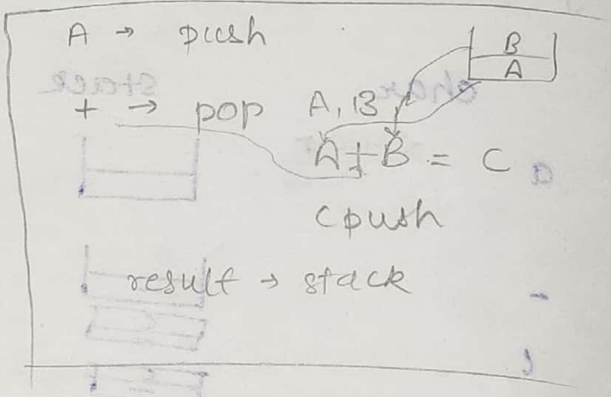
we don't insert (,) in postfix

24 Evaluation of Postfix expression →

① only stack is used.
② operand stack

① If char is ~~operator~~ operand, PUSH into stack.
② If char is operator, pop top 2 operands from the stack, perform calculation, push the result back into stack.

After reading all the char ~~in~~ from postfix exp. STACK will be only having RESULT.



eg → 562 * +

char | stack

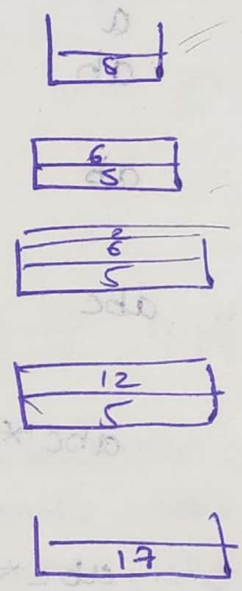
5

6

2

*

+



remove 2 * write 6 on left side put * & perform calculat

6*2 = 12

5 + 12 = 17

Result = 17

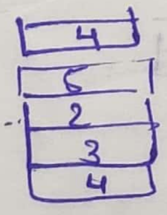
Eg. 4325 * - +

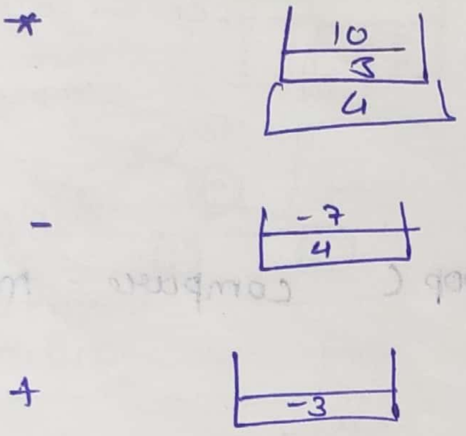
char | stack

4

3

2
5





$2 * 5 =$

$3 - 10 = -7$

$4 + -7 = -3$

Ans $= -3$

3) Balancing Symbols →

Balancing symbols
- ( ) > exp.
- [ ]
- { } → block of statements

→ Balancing every open symbol should have closed symbol

Algorithm →

① Read char from exp.

② If char is open symbol '(', '[', '{', push symbol into stack.

③ If char is closed symbol),], } , check if stack is empty
- if empty, exp is unbalanced
- if not, pop the symbol from stack and compare with the symbol which is read.
  - = if mathes, repeat the process
  - = if not, unbalanced.

④ After reading all the exp, stack not empty → unbalanced.

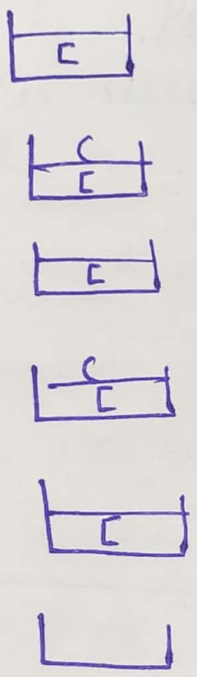
Example →

[(a+b)(a-b)]

24 Q.

open → push

$$\begin{tikzcd}[column sep=small] & \text{empty} \arrow[r, dash] & \text{unbalanced} \\ \text{close} \arrow[ur, dash] \arrow[dr, dash] & & \\ & \text{not empty} \arrow[r, dash] & \text{pop compare repeat} \end{tikzcd}$$



[

(

) pop (compare - matches

(

) pop (compare - matches

] pop [compare matches

BALANCED EXPRESSION

Use of stack

→ To reverse a word

→ application