CS & IT ENGINEERING

Data Structures

Stack and Queues

Lecture No.- 03

By- Pankaj Sharma Sir

Recap of Previous Lecture

Topic Stack and Queues Part - 02

- infix to postfix
- infix to prefix → without stack
- applications

Topics to be Covered

Topic Stack and Queues Part - 03

infix to prefix

Topic : Stack and Queues

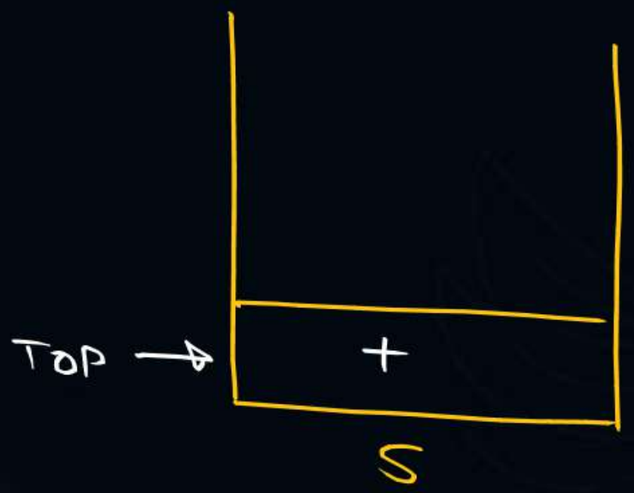
infix to prefix using stack

Ex 1 infix : 2 + 3

Reverse infix : 3 + 2

stack empty (Push)

O/P : 3



Topic : Stack and Queues

infix to prefix using stack

Ex 1 infix : 2 + 3

Reverse infix : 3 + 2

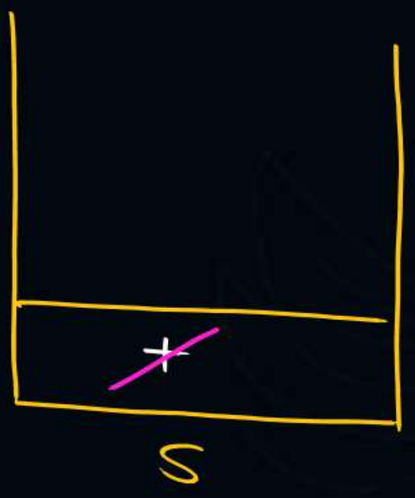
End

(Pop everything & add to o/p)

o/p : 3 2 +

Reverse : + 2 3

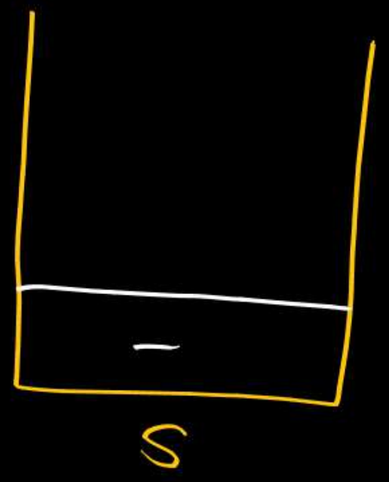
prefix



Ex2: infix : a + b − c

reverse of infix : c − b + a

O/P : c b



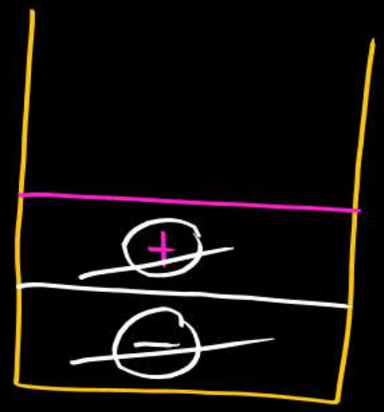
Ex2: infix : a + b − c

reverse of infix : c − b + a

O/P : c b a + −

End

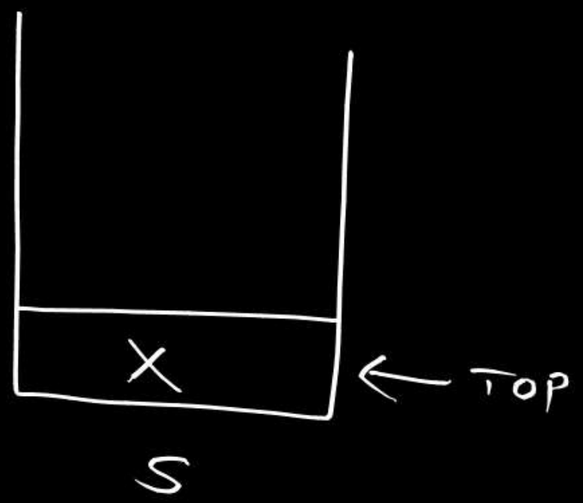
reverse : − + a b c



Ex 3: infix : 2 + 3 × 4

Reverse of infix : 4 × 3 + 2

o/p : 4 3



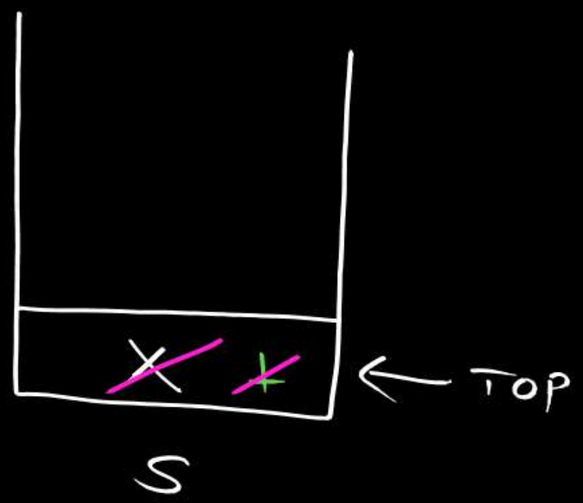
pop from stack & add to o/p

Ex 3: infix : 2 + 3 × 4

Reverse of infix : 4 × 3 + 2

o/p : 4 3 × 2 +

Reverse o/p : + 2 × 3 4

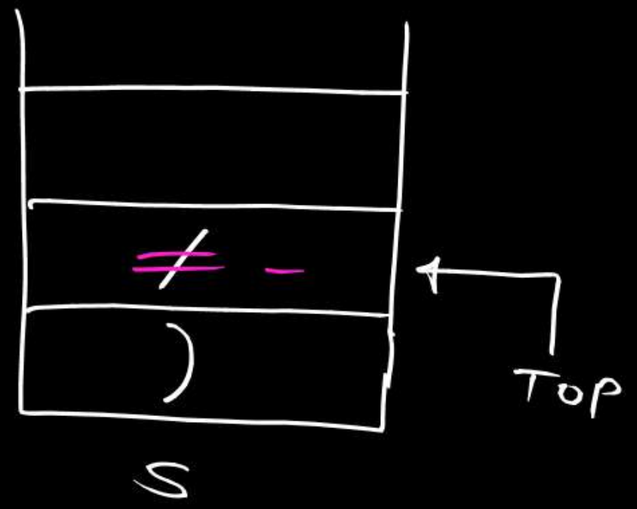


pop from stack & add to o/p

Ex4: infix : $a+(b\times c-d/e)$

Reverse infix :) e / d − c × b (+ a

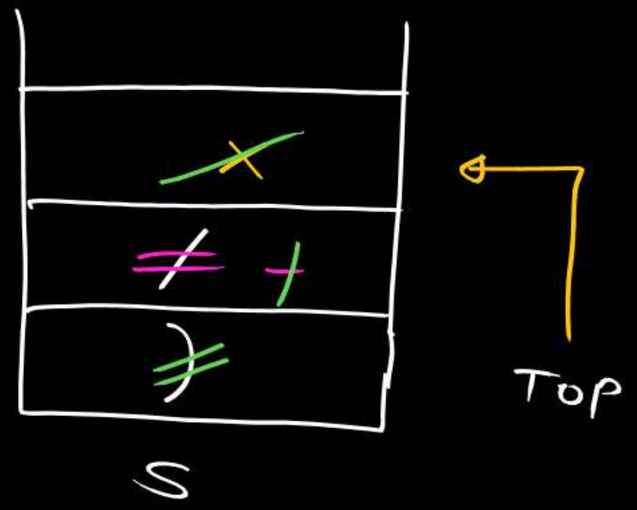
o/p: e d /



Ex4: infix : $a+(b\times c-d/e)$

Reverse infix :) e / d − c × b (+ a

o/p: e d / c b × −



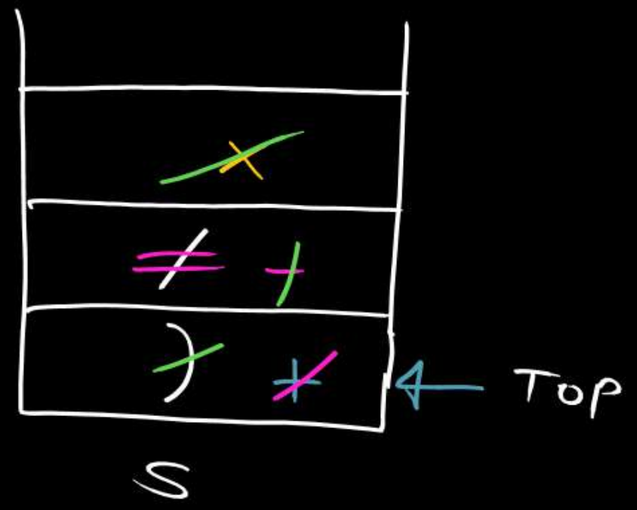
Ex4: infix : $a+(b\times c-d/e)$

Reverse infix :) e / d − c × b (+ a

End

o/p: e d / c b × − a +

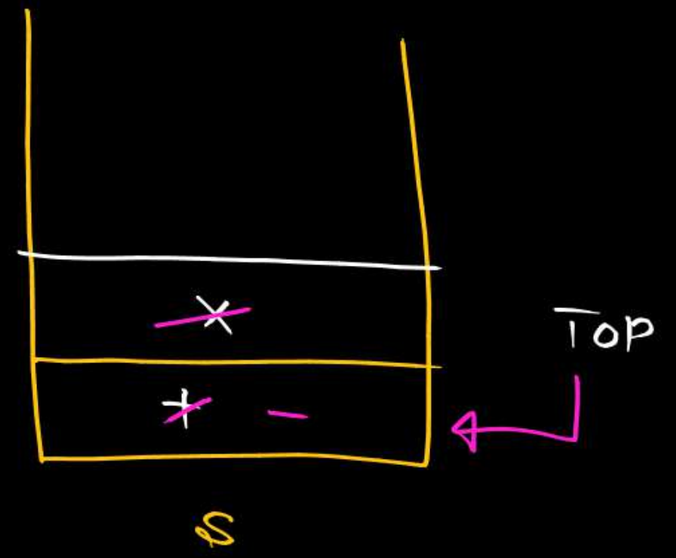
reverse : + a − × b c / d e



using stack

Q/ infix : a + b x c - d / e ^ f ^ g + h

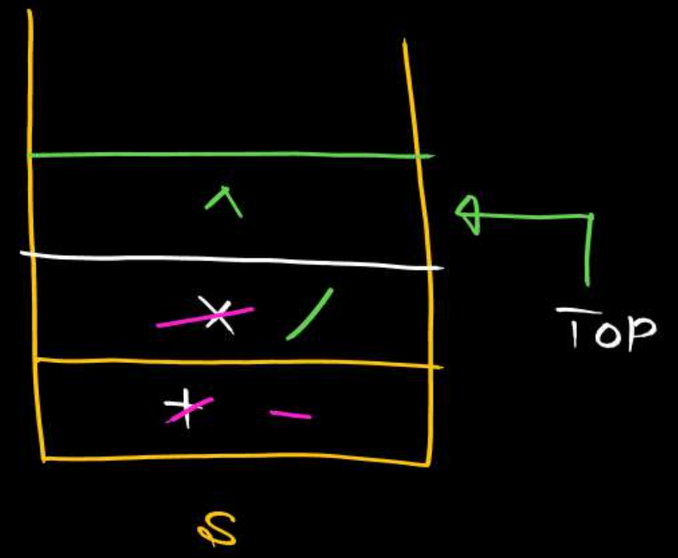
Postfix : a b c x +



Q/ infix : a + b x c - d / e ^ f ^ g + h

using stack

Postfix : a b c x + d e



Q/ infix : a + b x c - d / e ^ f ^ g + h

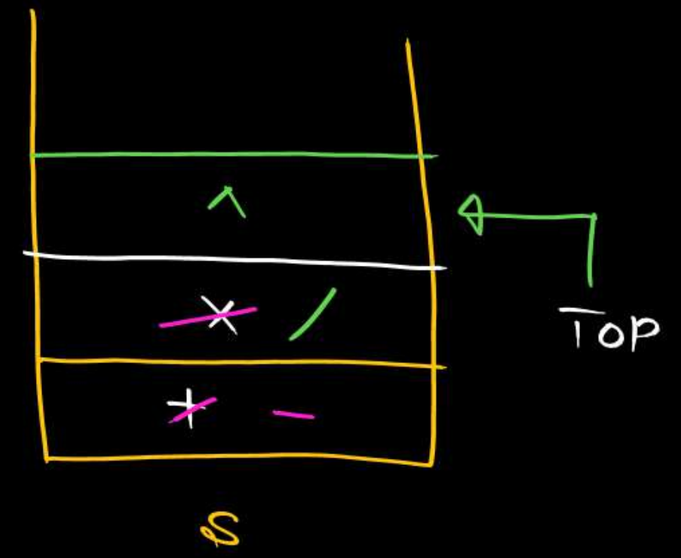
Postfix : a b c x + d e f

using stack

^ R to L

push

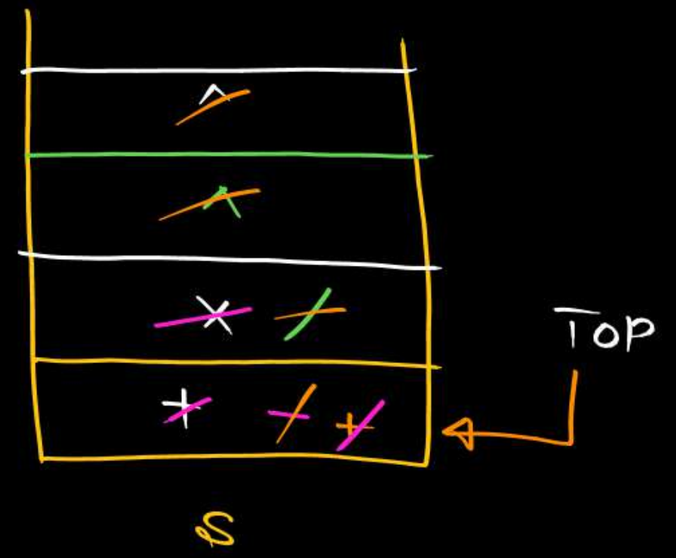
e ^ f ^ g



using stack

Q/ infix : a + b x c - d / e ^ f ^ g + h

Postfix : a b c x + d e f g ^ ^ / - h +



infix to prefix using stack

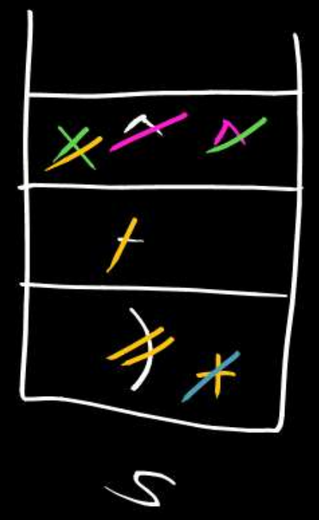
Q/ infix : A + (B x C ^ D ^ E - F)

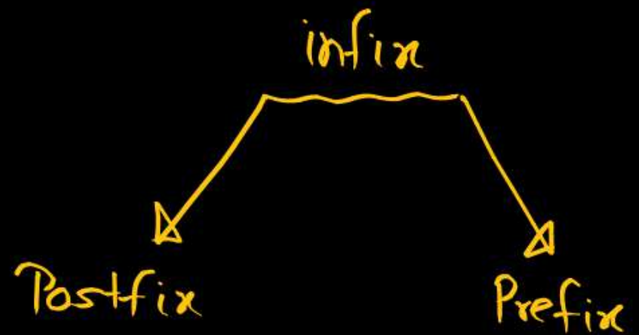
reverse infix :) F - E ^ D ^ C x B (+ A

End

O/P : F E D ^ C ^ B x - A +

Reverse O/P : + A - x B ^ C ^ D E F

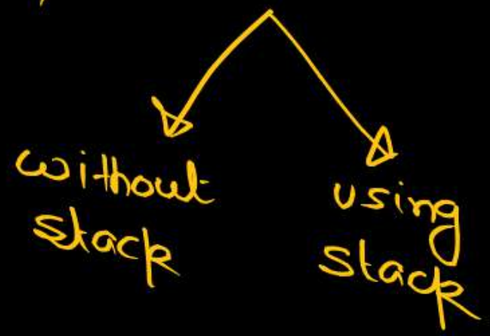




① infix to postfix:

stack : operators

② postfix evaluation

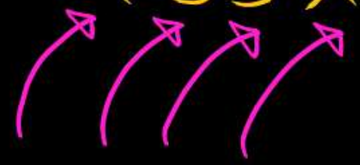


Ex1

infix : 2+3x5

Postfix : 2 3 5 X +

Stack → operands



① pop two elem. from stack

1st pop → A

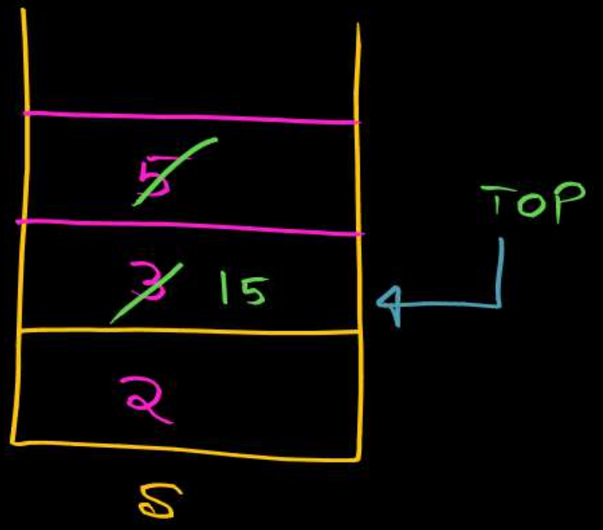
2nd pop → B

② perform B opstr A

③ push result on stack

A → 5

B → 3



pop order

5, 3

3 X 5

= 15

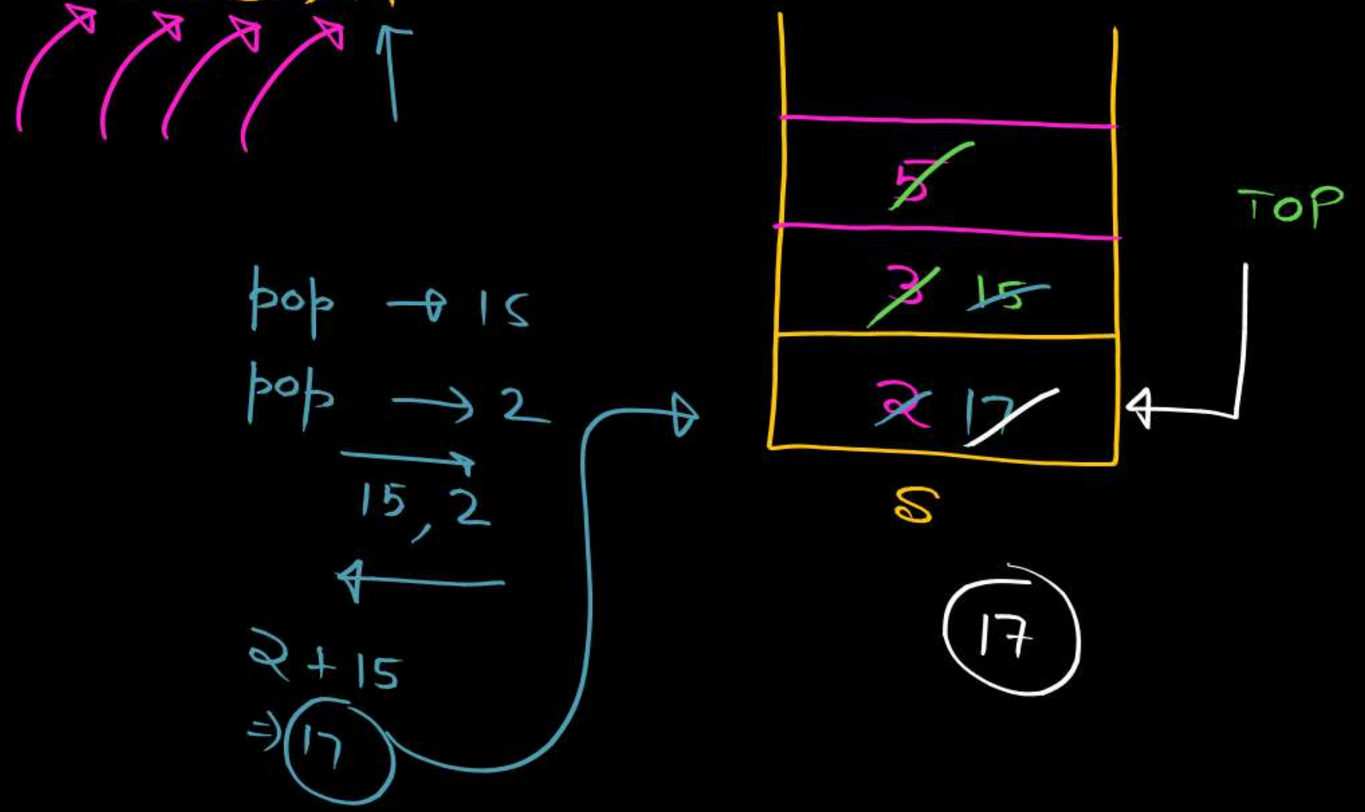
Ex1

infix : 2+3x5

Postfix : 2 3 5 x +

End

Stack → operands



$2 + [3\,4\,X] - [6\,2\,/]$

Ex2: infix : $2 + 3 \times 4 - 6/2$

Postfix : $2\ 3\ 4\ X\ +\ 6\ 2\ /\ -$

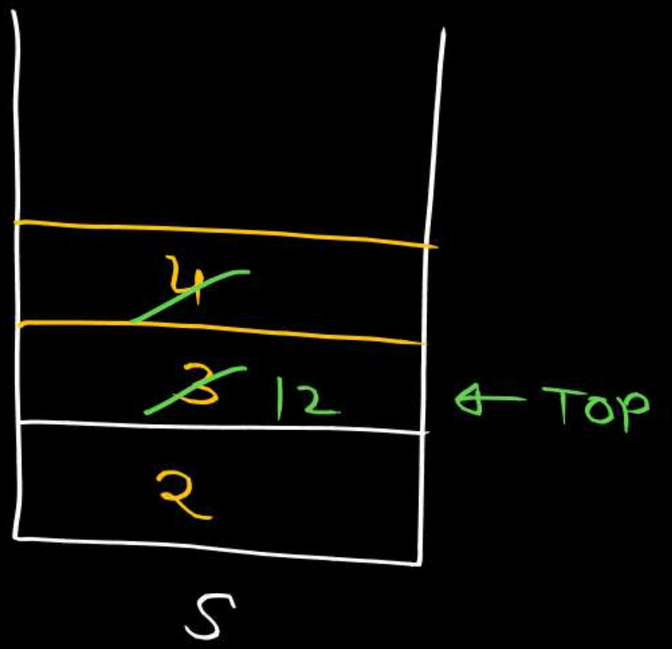
X → (i) Pop 4

(ii) Pop 3

4, 3

3 X 4

Push 12



$2 + [3\,4\,X] - [6\,2\,/]$

Ex2: infix : $2 + 3 \times 4 - 6/2$

Postfix : $2\ 3\ 4\ X\ +\ 6\ 2\ /\ -$

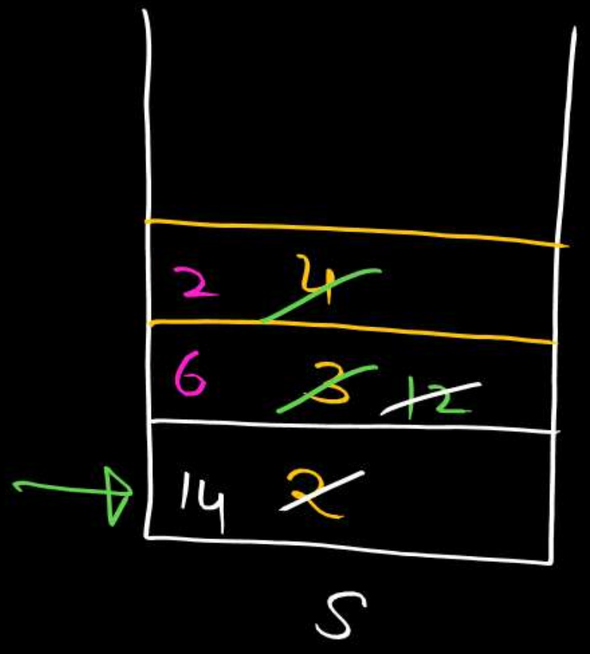
Pop 12

pop 2

12, 2

$2 + 12$

(14)



$2 + [3\,4\,\times] - [6\,2\,/]$

Ex2: infix : $2 + 3 \times 4 - 6/2$

Postfix : $2\ 3\ 4\ \times\ +\ 6\ 2\ /\ -$

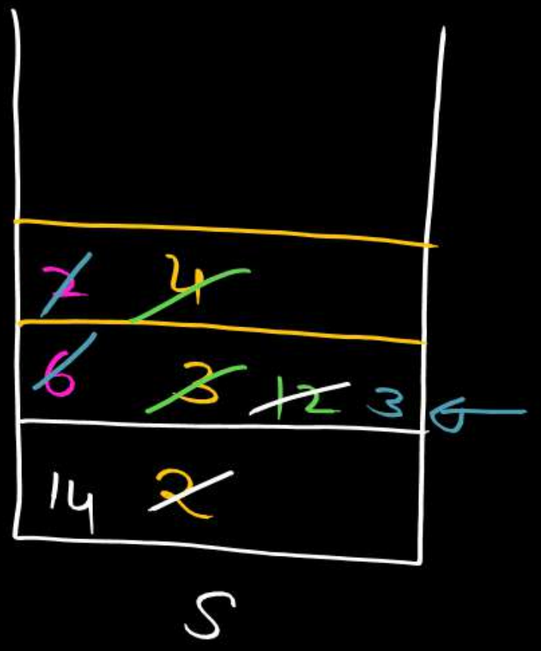
(ii) pop 2

(iii) pop 6

2, 6

$6/2$

$= 3$



$2 + [3\,4\,X] - [6\,2\,/]$

Ex2: infix : $2 + 3 \times 4 - 6/2$

Postfix : $2\ 3\ 4\ X\ +\ 6\ 2\ /\ -$

End

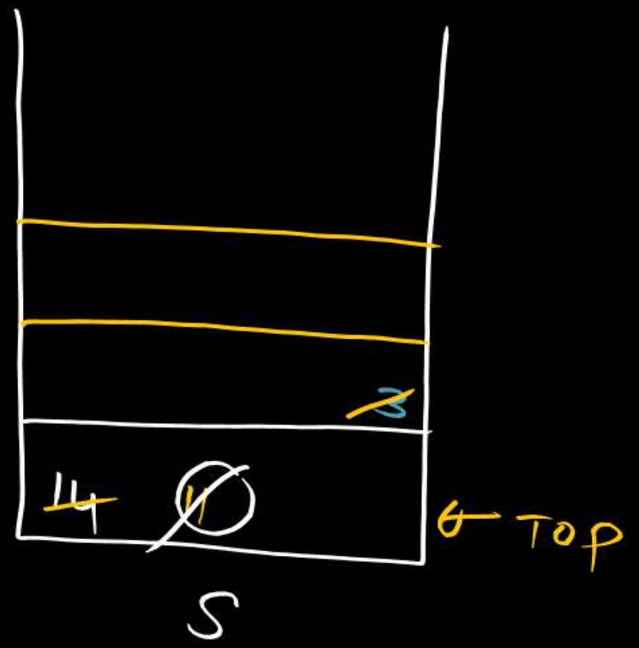
pop 3

pop 14

3 | 14

14 - 3

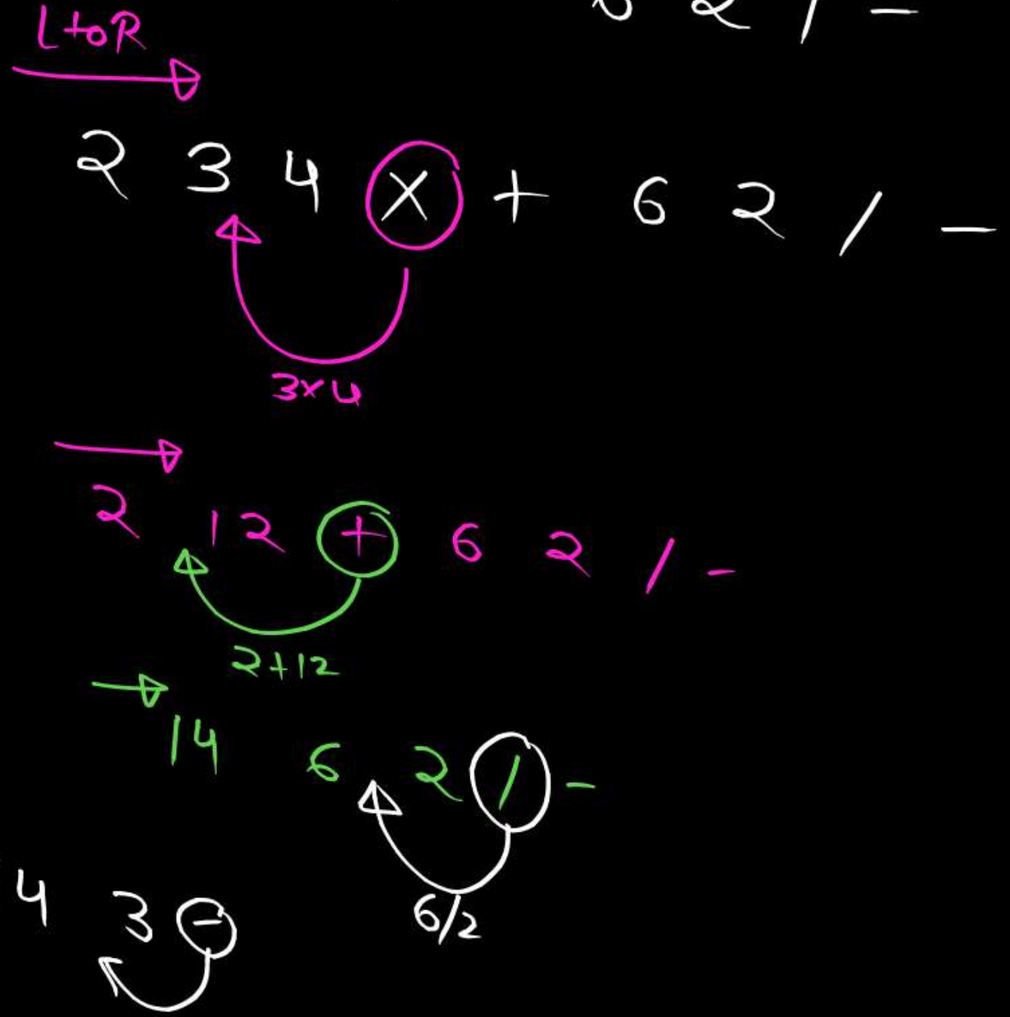
= (11)



Ans: 11

Ex2:

Postfix : 2 3 4 × + 6 2 / −



Prefix Evaluation

infix : 2 + 3 × 5

Prefix : + 2 × 3 5

Prefix Evaluation

Prefix : +2X35

Reverse Prefix : 5 3 X 2 +

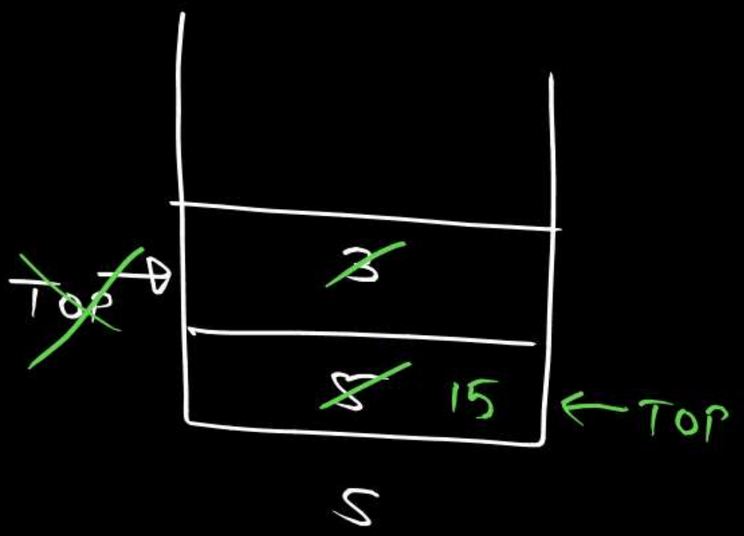
(i) Pop 3

(ii) Pop 5

3, 5

3 X 5

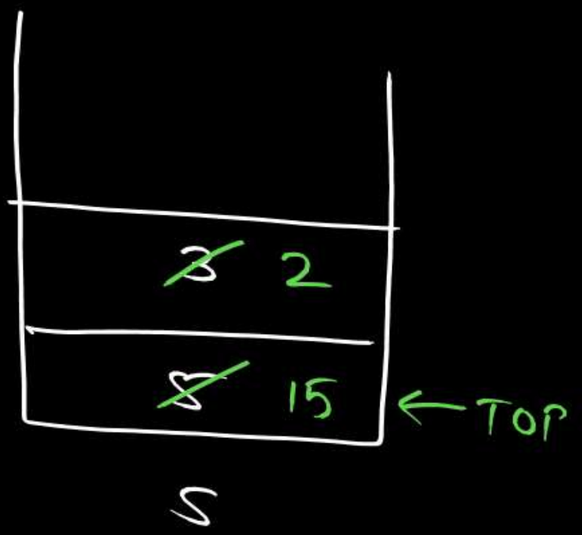
Push 15



Prefix Evaluation

Prefix : +2X35

Reverse Prefix : 5 3 X 2 +



Prefix Evaluation

Prefix : +2X35

Reverse Prefix : 5 3 X 2 +

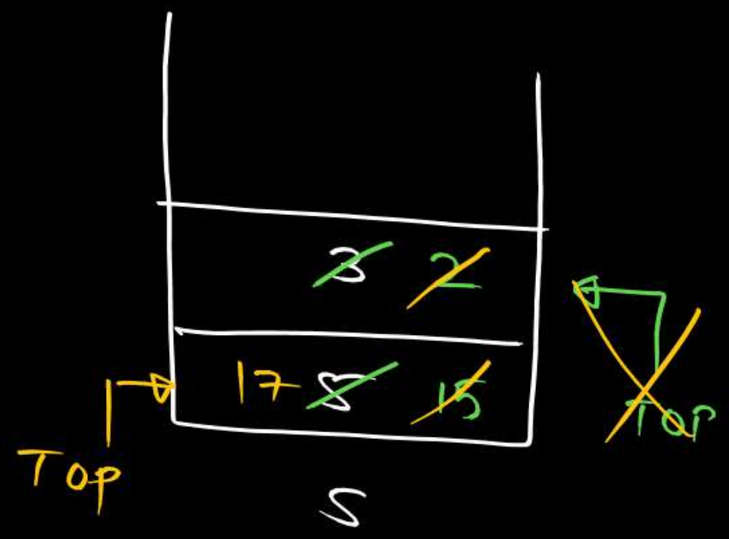
pop 2

pop 15

2, 15

2+15

Push 17



Prefix Evaluation

Priyanshi

Prefix : + 2 X 3 5

Reverse Prefix : 5 3 X 2 +

End

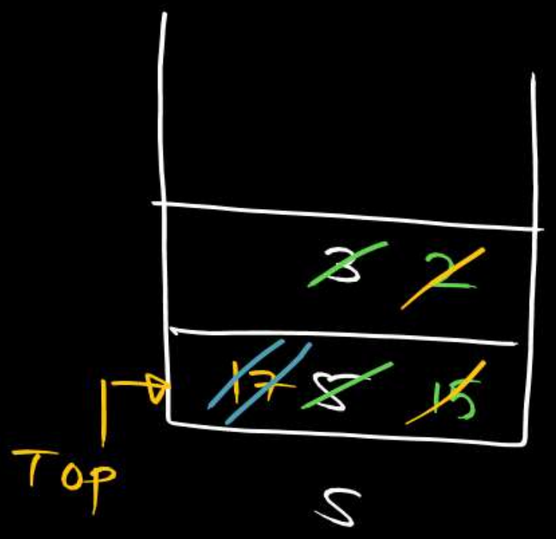
pop 2

pop 15

2, 15

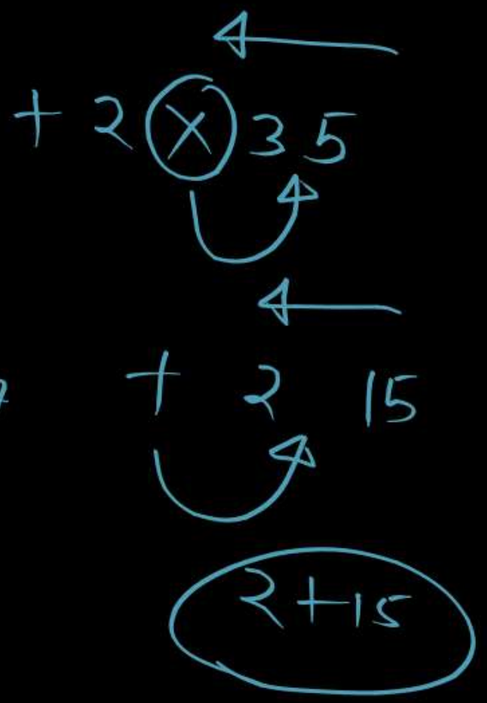
2+15

Push 17



Ans

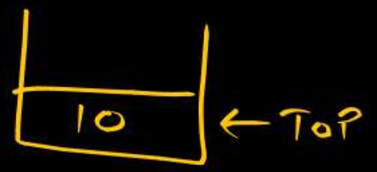
17



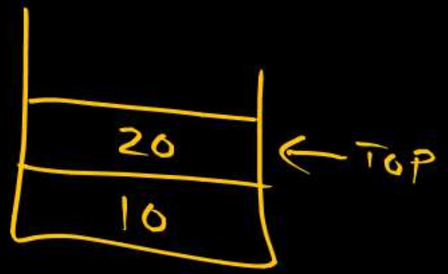
Push → O(1)
Pop →
} constant time

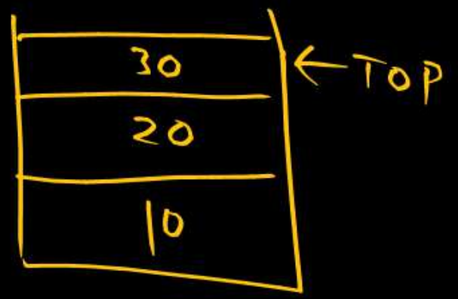
(i) Push(10)

(ii) Push(20)

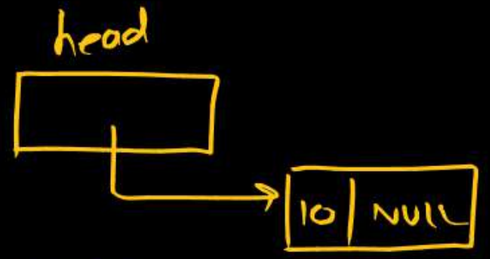


(iii) Push(30)

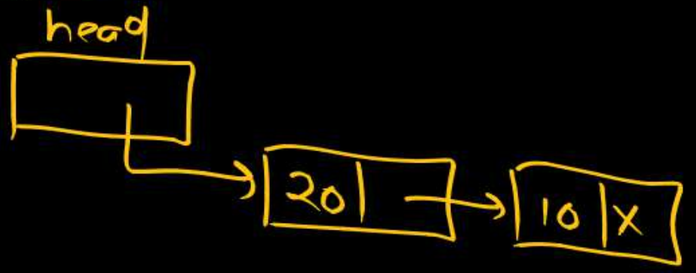


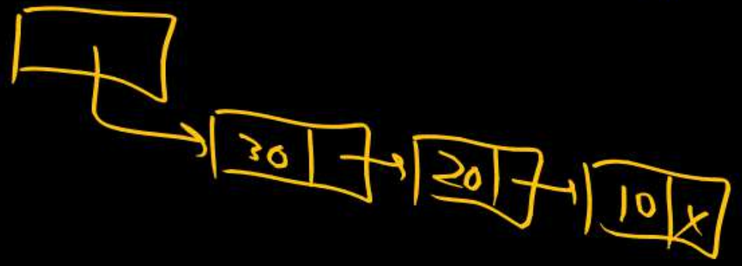


Stack using L.L



Insert at begin

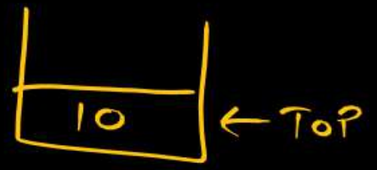




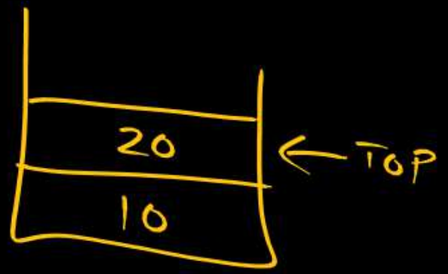
Push → O(1)
Pop →
} constant time

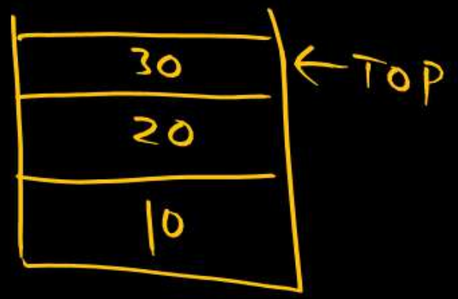
(i) Push(10)

(ii) Push(20)

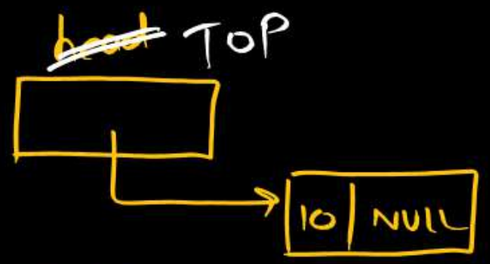


(iii) Push(30)



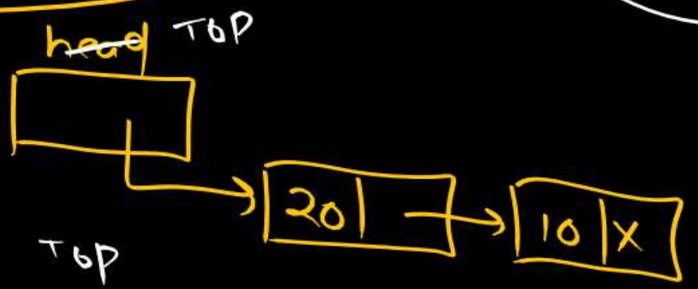


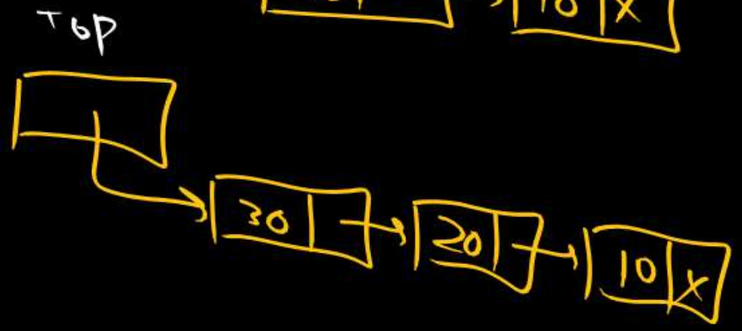
Stack using L.L



Insert at begin

delete at begin





Q/ The result of eval the postfix expression:

10 5 + 60 6 / * 8 - is

A) 284

B) 213

C) 142 ✓

D) 71

15 60 6 / * 8 -

15 10 * 8 -

150 8 -

142

Q The best data structure to check whether an arith. exp. has bal. paranthesis is ___

A) Queue

B) stack

C) Tree

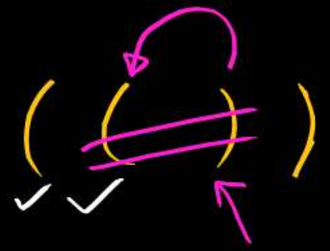
D) list

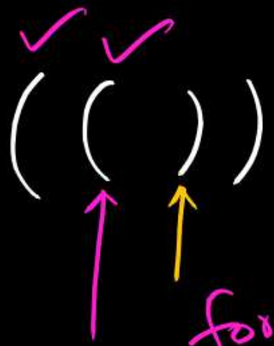
$(a+b) \times (c+d)$

( )( ) ⌙

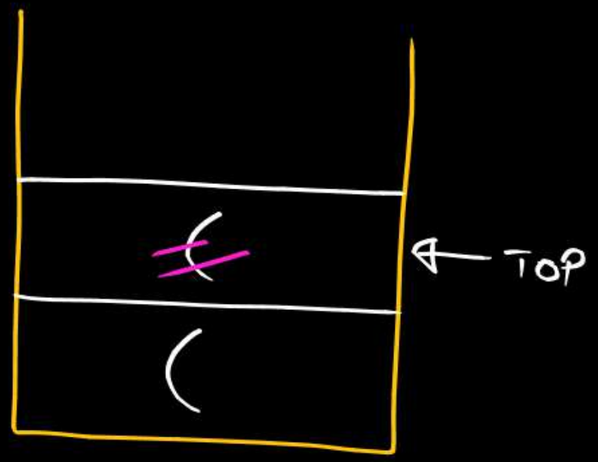
(( )) ⌙

( )( )(( )) ⌙





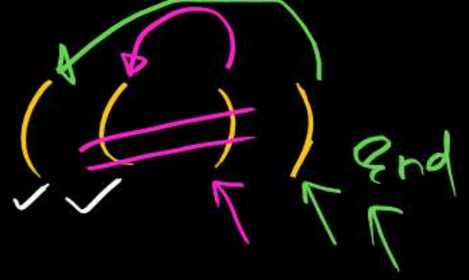
for Every right bar

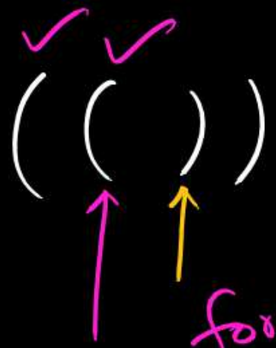


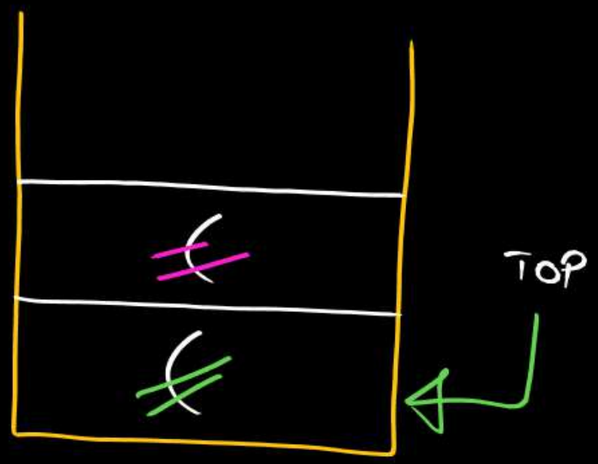
TOP

Most recently occurred

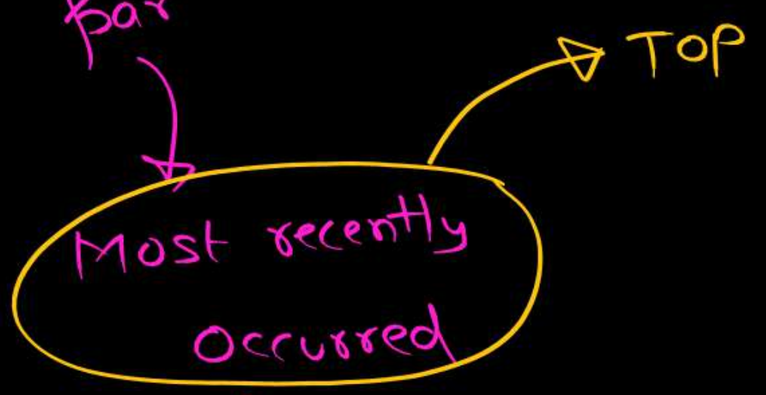
left parenthesis



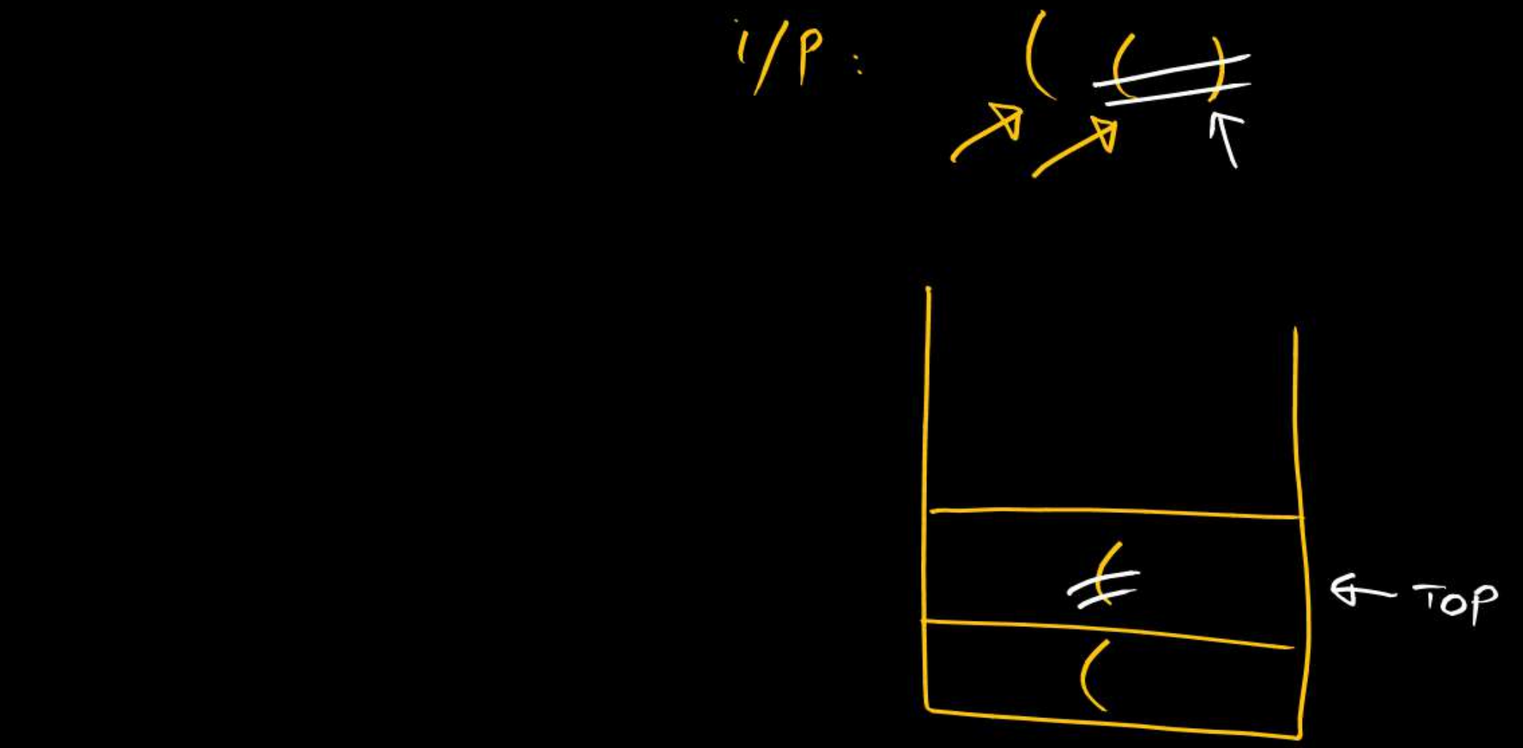




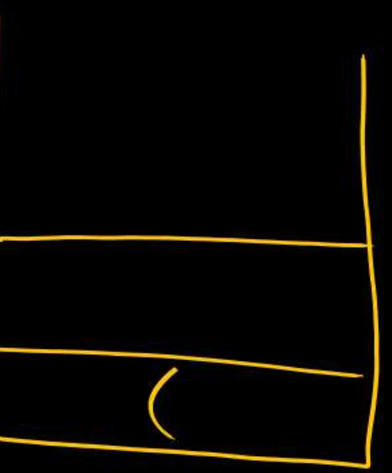
Valid



left parenthesis


i/p :
TOP

i/p : (( ))



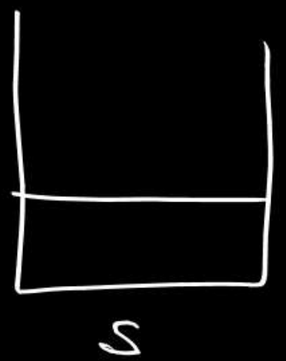
Invalid

i/p :) ( )

right Bar

⇓

stack is Empty



S

Invalid

Q Which of the foll. is essential to convert infix exp to postfix exp. efficiently?

A) Operator stack

B) Operand stack

C) An operator and an operand stack.

D) A parse tree

Q Which of the following permutations can be obtained in the o/p (in same order) using a stack assuming that the i/p seq is 1,2,3,4,5 in that order.

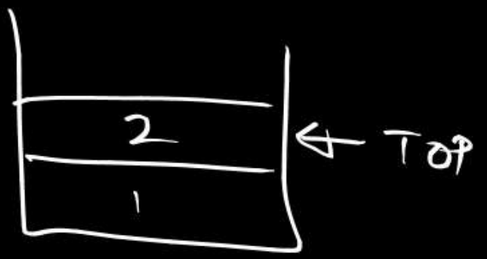
A) 3, 4, 5, 1, 2 ✗

B) 3, 4, 5, 2, 1 ✓

C) 1, 5, 2, 3, 4

D) 5, 4, 3, 1, 2

Push 1
Push 2
Push 3
Pop
Push 4
Pop
Push 5
Pop



Q Which of the following permutations can be obtained in the o/p (in same order) using a stack assuming that the i/p seq is 1,2,3,4,5 in that order.

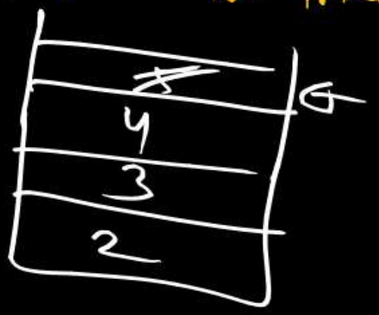
A) 3, 4, 5, 1, 2 ✗

B) 3, 4, 5, 2, 1 ✓

C) 1, 5, 2, 3, 4 ✗

D) 5, 4, 3, 1, 2 ✗

Push 1
Pop()
Push 2
" 3
" 4
" 5
Pop



Questions

THANK - YOU