unacademy

Stack and Queue - Part III

Course on Data Structure

Pankaj Sharma • Lesson 16 • July 26, 2024

unacademy

CS & IT Engineering

Data Structure

Stack & Queue

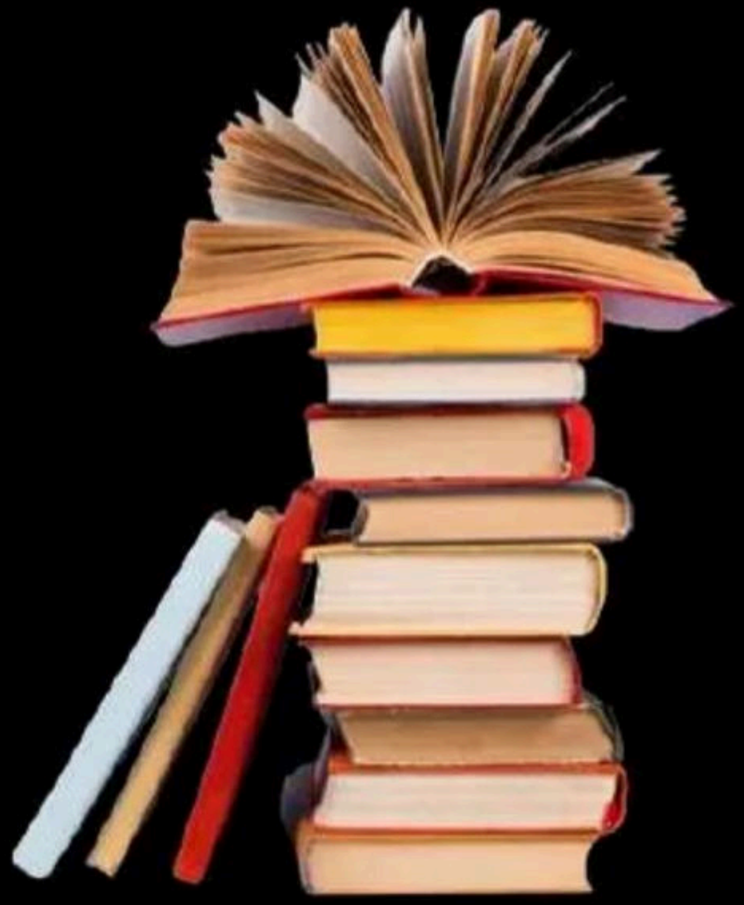
Lecture Number- 16

By- Pankaj Sir

unacademy

Topics

to be covered



1. Stack

1,2,3

stack permutation

$n=3$

1) 1,2,3
2) 1,3,2
3) 2,1,3
4) 2,3,1
5) 3,1,2
6) 3,2,1

} 3! permutation.

(i) Insertion order is fixed

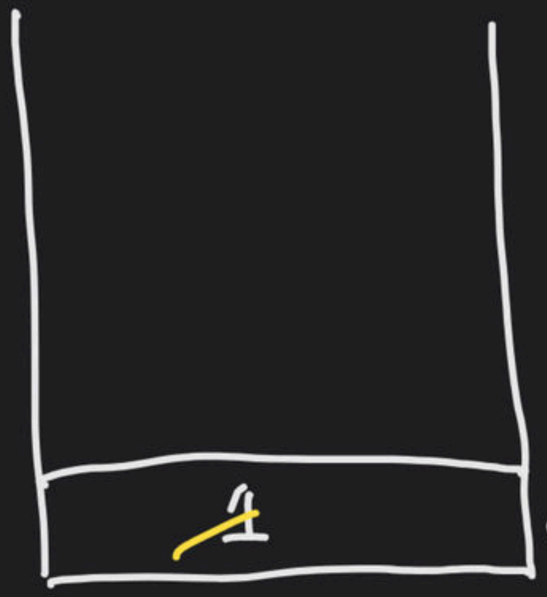
i.e. 1,2,3 ⟶

but you can pop() any time.

unacademy

Insertion order : 1, 2, 3

1.) Pop-order: 1, 2, 3



Push(1)

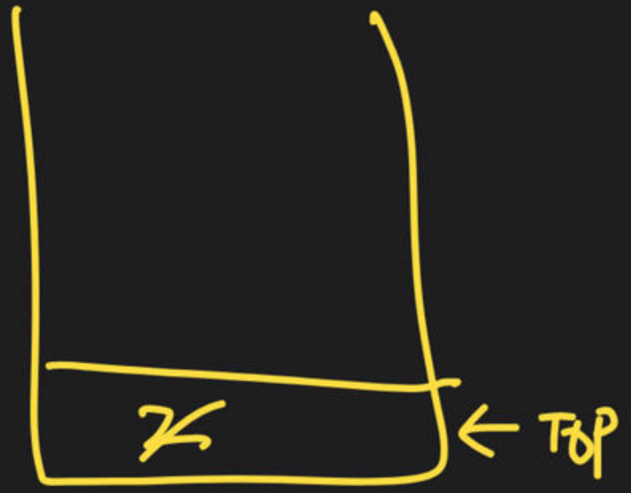
Pop() → 1

Push(2)

Pop() → 2

Push(3)

Pop()



1.)

pop-order

stack permutation.

Push(1)
Pop()
Push(2)
Pop()
Push(3)
Pop()

2) 1, 3, 2 (pop - seq.)

1, 2, 3

a)

3) 2, 1, 3

a) Push(1)

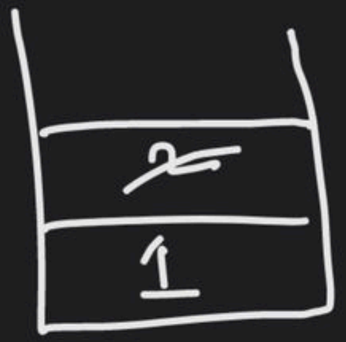
b) Push(2)

c) Pop()

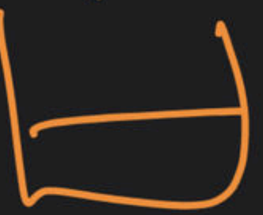
d) Pop()

e) Push(3)

f) Pop()



Pop()



Push(3)

3

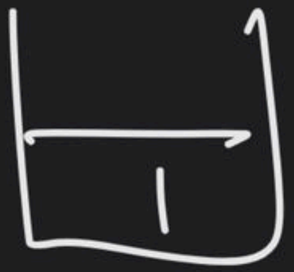
Pop()

4) 2, 3, 1 → valid pop seq.

Push(1)

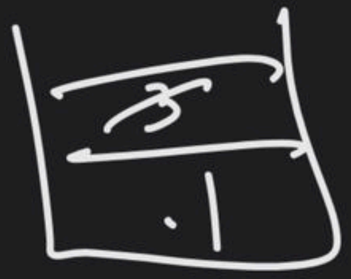
Push(2)

Pop() → 2



Push(3)

Pop()



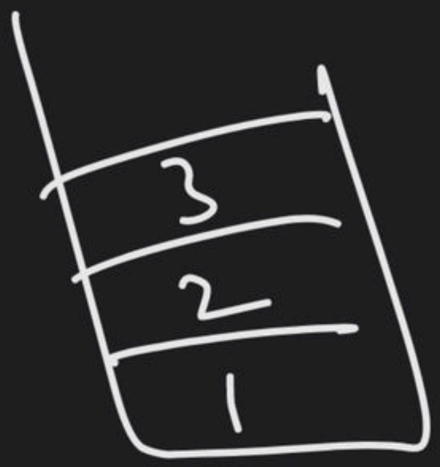
3

Pop()

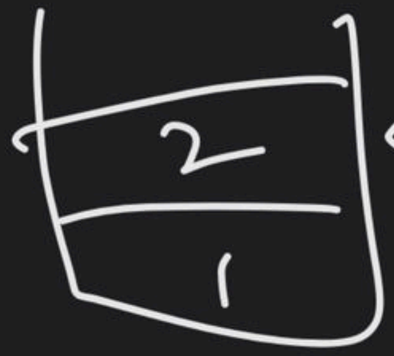
1

5) 3, 1, 2 → not a valid stack permutation

Push(1)
Push(2)
Push(3)
Pop()



→



6) 3, 2, 1

Push(1)

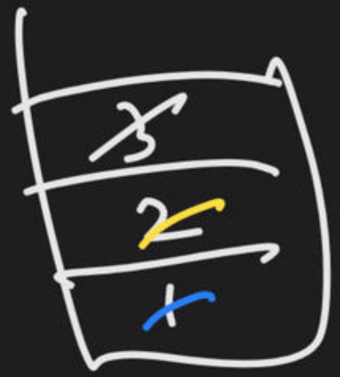
Push(2)

Push(3)

Pop()

Pop()

Pop()



1, 2, 3

Total no. of permutation = 6

Out of 6 possible permutation $\Rightarrow$ 5 are valid stack permutations.

Total valid stack permutation = $\frac{^{2n}C_n}{n+1}$ Catalyn number

unacademy

Infix, prefix, postfix

operator

infix: 2 + 3

operand operand

Postfix: Operators comes after operands

2 3 +

Prefix: operator comes before operands

+ 2 3

Why Postfix ? Eyes

i/p: 3 + 4 x 6 / 3 ^ 2

left to right

3 + 4 x 6 / 9

Multiple scans

T.C:

unacademy

Applications of stack.

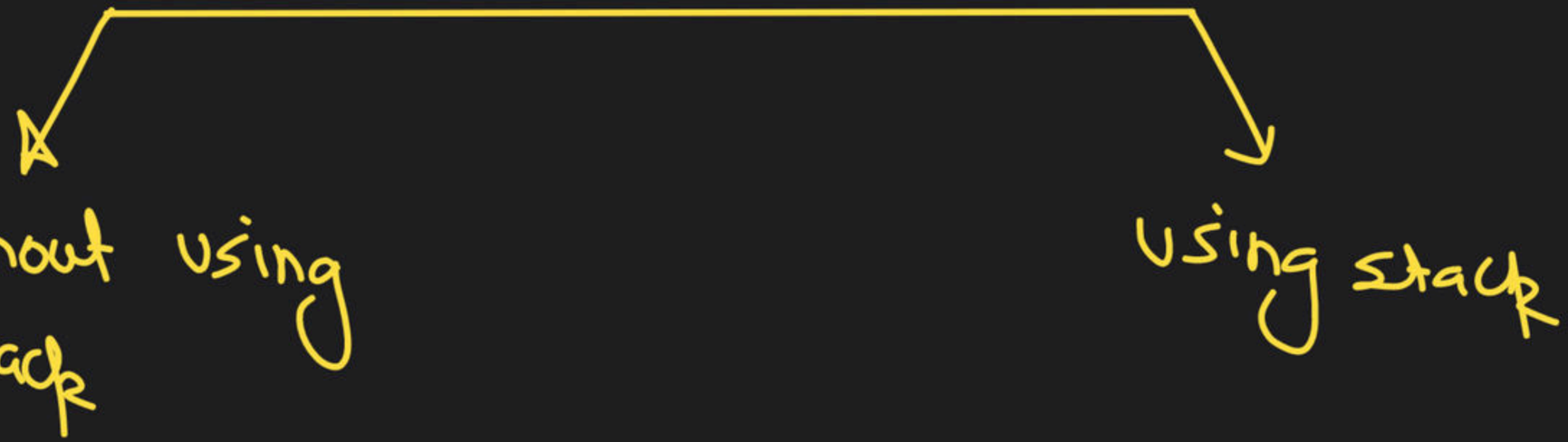
1.) infix ⟶ Postfix

2.) Evaluate postfix Expression

infix Expression Evaluated ?

Priority
Associativity.

infix to postfix Conversion



Ex:-

I/P : 2 + 3 X 5

postfix

2 + [3 5 X]

operand1

operand2

Postfix : 2 3 5 X +

unacademy

Ex 2:

i/p: 3+5 X 6/2^4

⇓

i/p: 3+ 5 X 6 / [2 4^]

3+ [5 6 X] / [2 4^]
op1 op2

op1

3+ [op2 / 5 6 X 2 4^ /]

⇒ 3 5 6 X 2 4^ / +

Postfix

Theoritical

① ^ R to L
② X , /
③ + - L to R

↑

Eg 3.

Infix : $(a+b) \times c/d - e \wedge f \wedge g/h$

$\wedge$ R to L

$\times$ // L to R

$+$, $-$ L to R

$[ab+] \times c/d - e \wedge f \wedge g/h$

$[ab+] \times c/d - e \wedge [fg\wedge]/h$

$[ab+] \times c/d - [efg\wedge\wedge]/h$

$[ab+c\times]/d - [efg\wedge\wedge]/h$

$[ab+c\times d/] - [efg\wedge\wedge]/h$

$[ab+c\times d/] - [efg\wedge\wedge h/]$

Postfix : $ab+c\times d/efg\wedge\wedge h/-$

unacademy

Infix to postfix using stack

infix : 2 + 3 →

postfix : 2 3 +

prefix : + 2 3

infix : 2 + 3 x 4 →

postfix : 2 3 4 x +

prefix : + 2 x 3 4

Ex 1.

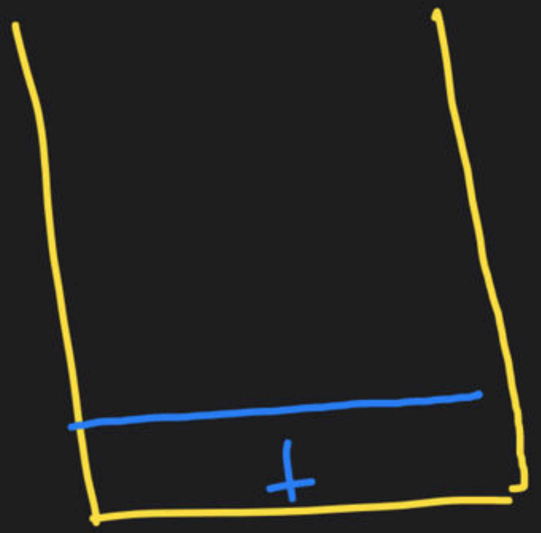
infix : 2+3

o/p : 2

Ex1.

infix : 2+3

o/p : 2



scanned element is operator

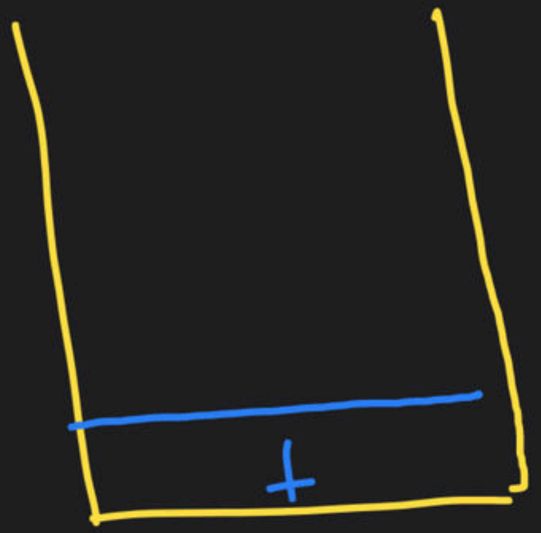
⇒ and stack is Empty

Push

Ex1.

infix: 2+3

o/p: 23



scanned element is operator

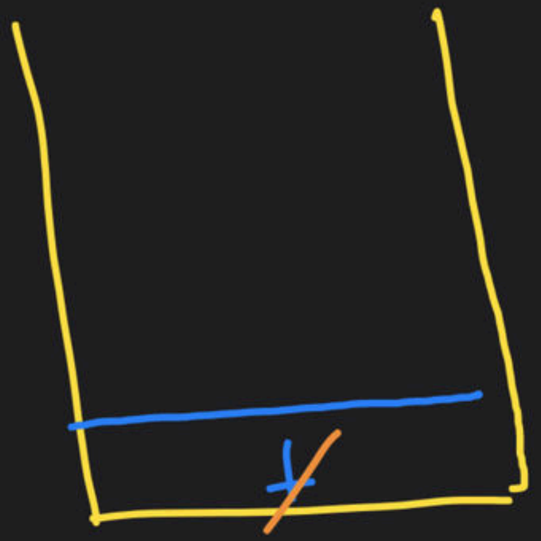
⇒ and stack is Empty

Push

Ex 1

infix: 2+3 End

o/p: 23



scanned element is operator

⇒ and stack is Empty

Push

Pop every thing from stack and add to o/p (one by one)

o/p: 23+

unacademy

Ex 2.

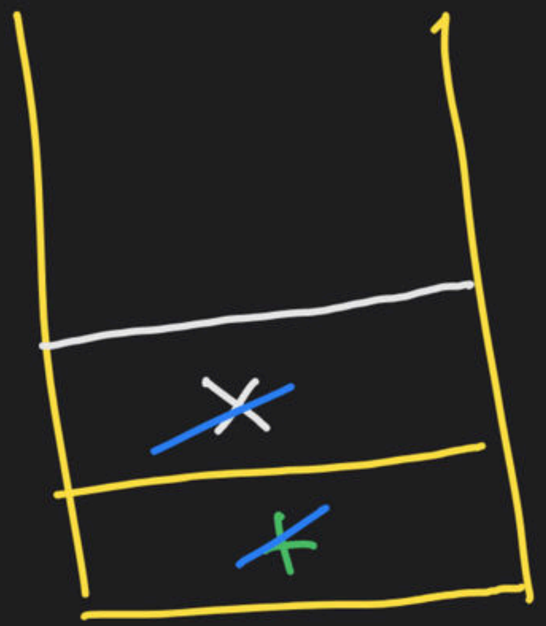
I/P : 2 + 3 × 4

O/P : 23

Ex 2.

I/P : 2 + 3 X 4 End

O/P : 2 3 4 X +

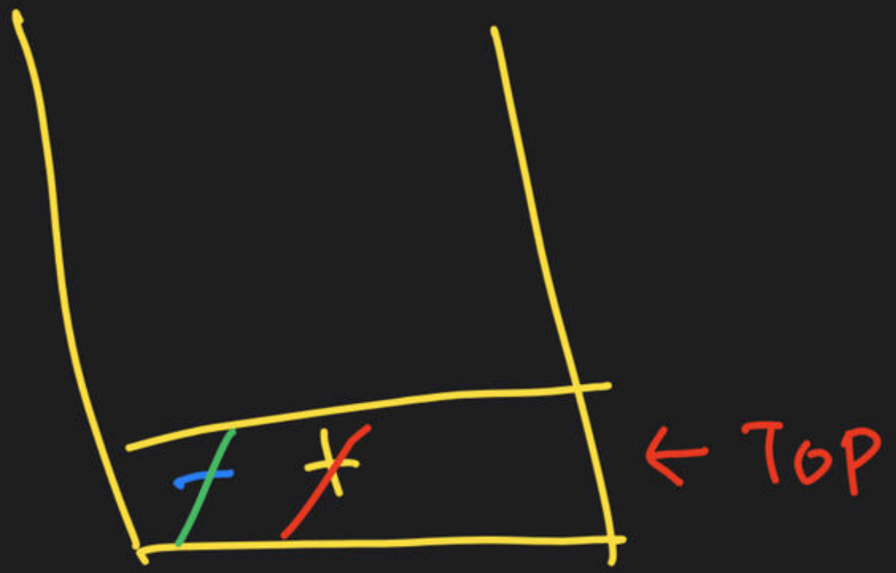


Ex3

i/p: a + b - c end

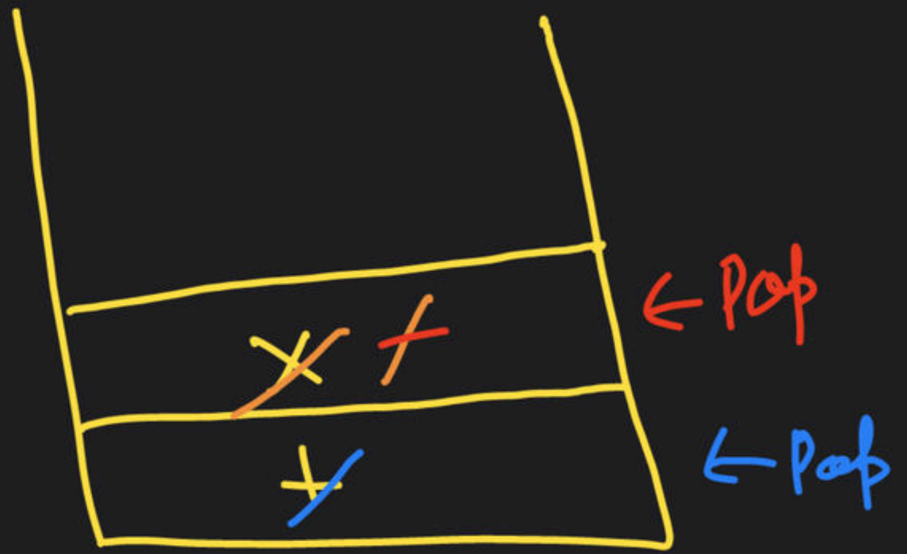
o/p : ab + c -

a + b - c



Ex 4 i/p: a + b x c / d end

o/p : a b c x d / +



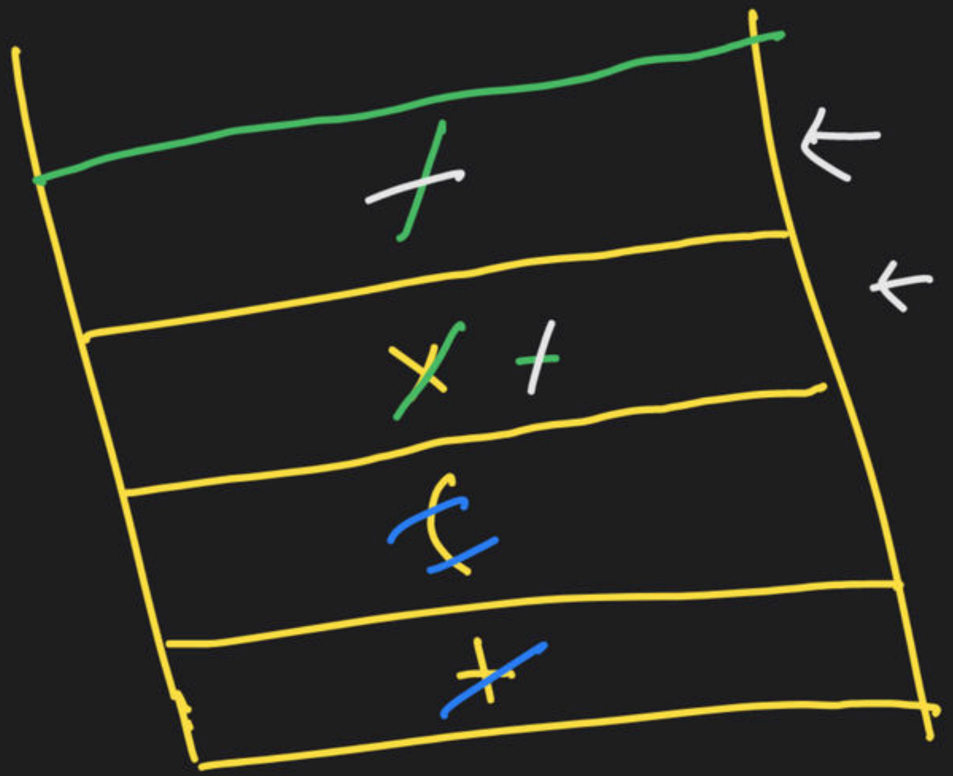
unacademy

i/p: 2 + (3 x 4 - 6 / 2) ← End

left (⇒ Push

o/p: 2 3 4 x 6 2 / - +

) ⇒ repeatedly Pop & add to o/p (one by one) until (is evaluated



(⇒ Pop( ) & discard

Ex 6

→ new concept

unacademy

▲ 1 • Asked by Deepraj

sir postfix me ^ operator ko hamesha power operator samjhekya cos ^ to bitwise xor bhi hota hein na

unacademy

THANK YOU!

Here's to a cracking journey ahead!