CS & IT ENGINEERING

Data Structure

Stack and Queues

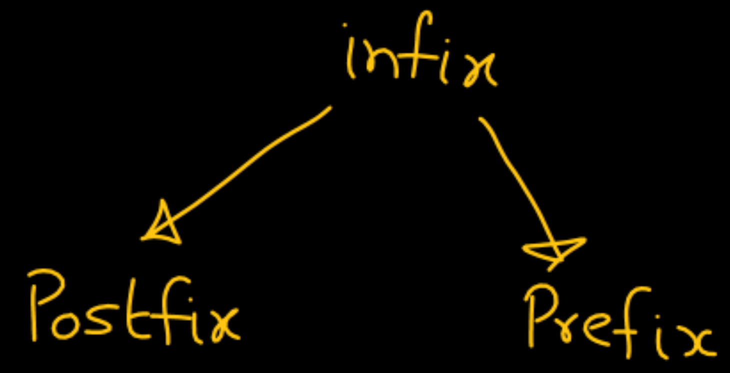
Chapter- 4

Lec- 04

By- Pankaj Sharma sir

TOPICS TO BE COVERED

- Stack-IV



infix

Postfix

Prefix

(i) Postfix evaluation

infix : $2 + 3 \times 5$

Postfix : $\underrightarrow{2\ 3\ 5 \times +}$

① infix → Postfix

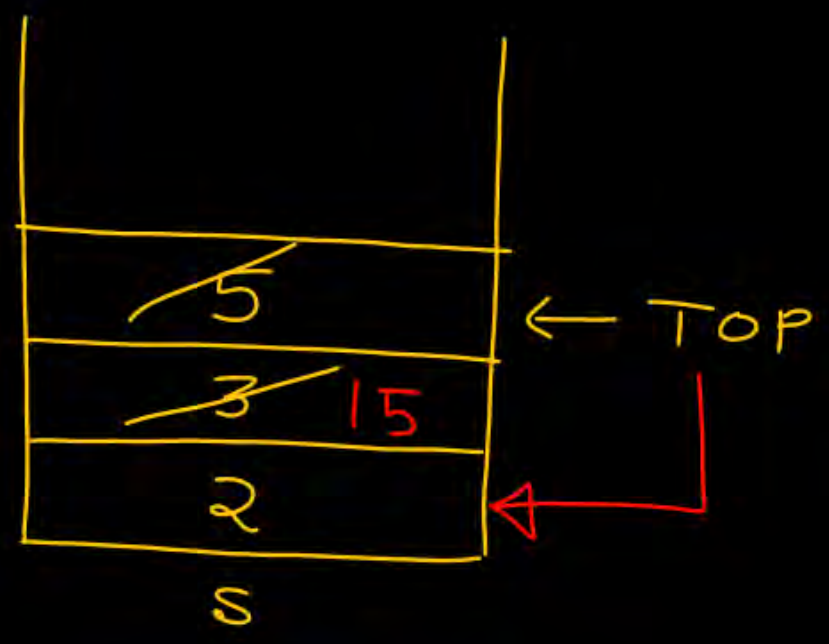
stack : operators

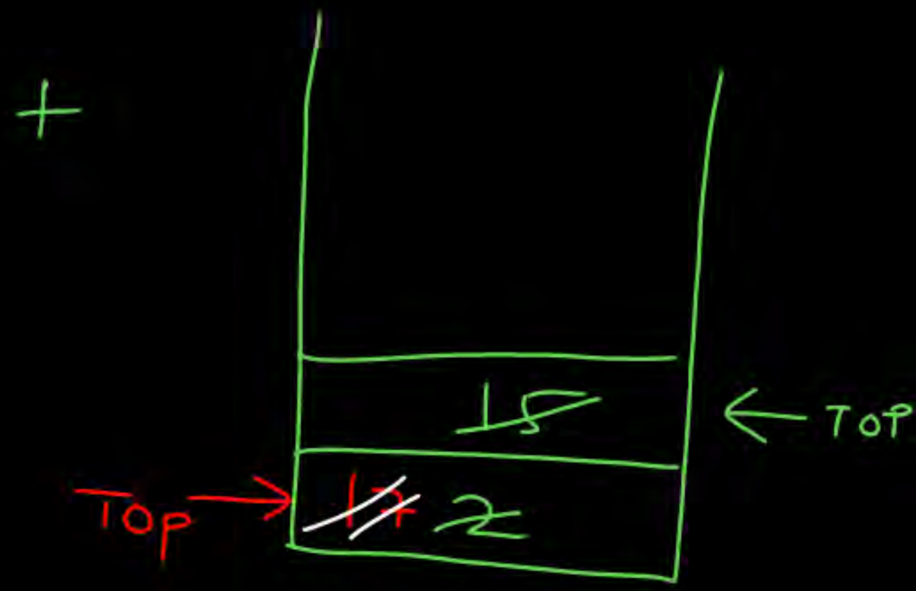
② Postfix eval.

⟹ stack : operand

Ex1

Postfix : 2 3 5 X + End





X → Pop 2 elements

A → 1st ele

B → 2nd ele

A = 5

B = 3

perform B X A & push the result on stack.

→ 5, 3 ←

3 × 5 = 15

17

A = 15

B = 2

→ 15, 2 ←

2+15 = 17

Ex2:

2+ 12-3

infix : $2 + \underline{3 \times 4} - 6/2$

Postfix : 2 3 4 x + 6 2 / − End

2+ [3 4 x] − [6 2 /]

[2 3 4 x +] − [6 2 /]

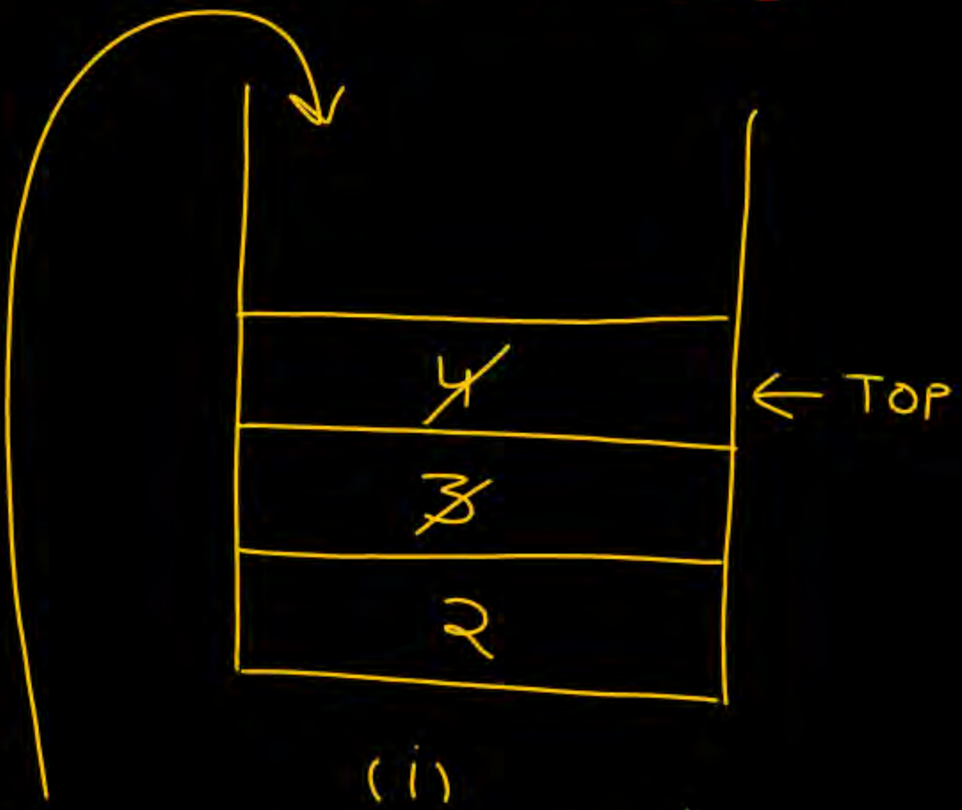
x

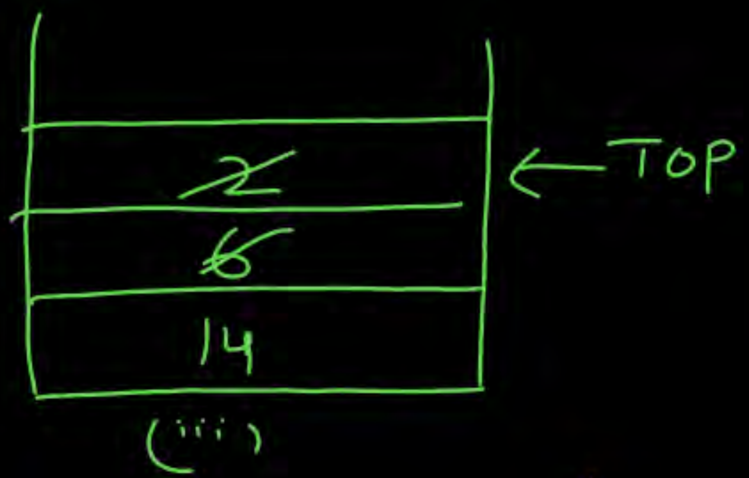
A = 4

B = 3

4,3

3x4 = 12





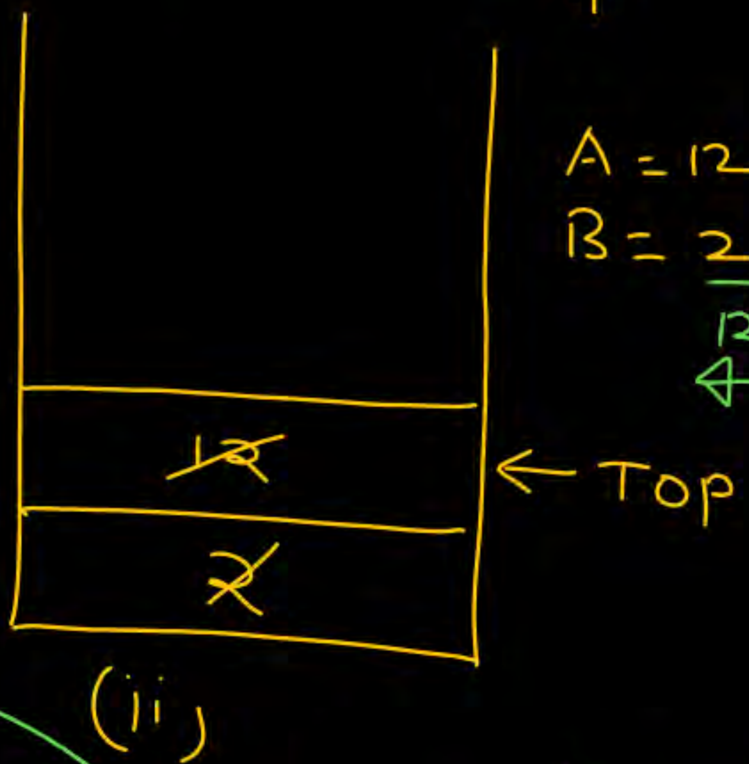
A = 2

B = 6

2,6

6/2 = 3

+



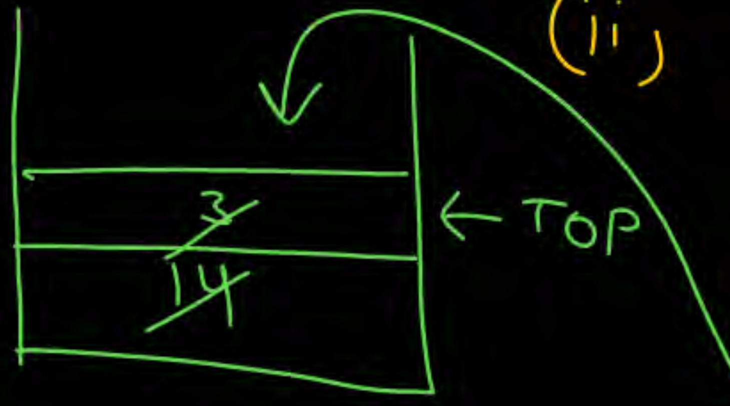
A = 12

B = 2

12,2

2+12 = 14

−

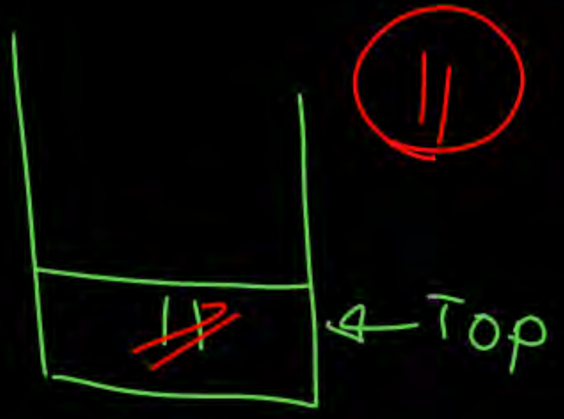


A = 3

B = 14

14-3 = 11

(iv)



$$\overrightarrow{2 \quad \boxed{3 \quad 4 \quad \times}} \quad + \quad 6 \quad 2 \quad / \quad -$$

$$\boxed{2 \quad 12 \quad +} \quad 6 \quad 2 \quad / \quad -$$

$$14 \quad \boxed{6 \quad 2 \quad /} \quad -$$

$$\boxed{14 \quad 3 \quad -}$$

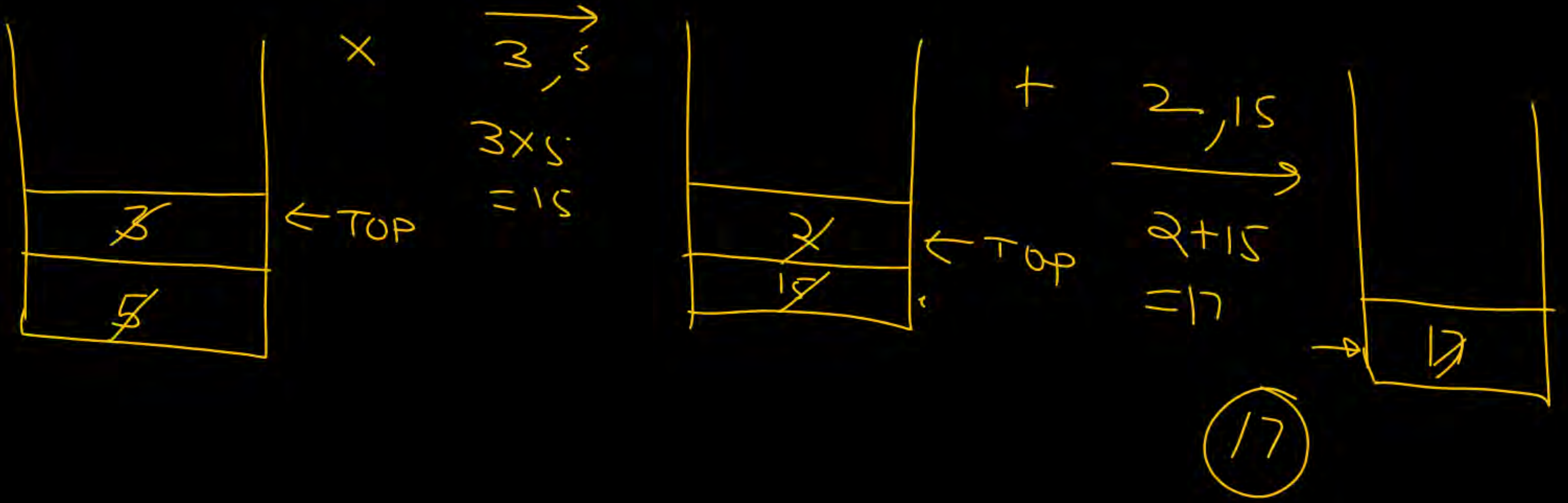
$$\boxed{11}$$

infix to prefix ✓

infix : 2 + 3 × 5

Prefix : + 2 × 3 5 (Prefix Evaluation)

reverse prefix : 5 3 × 2 + End



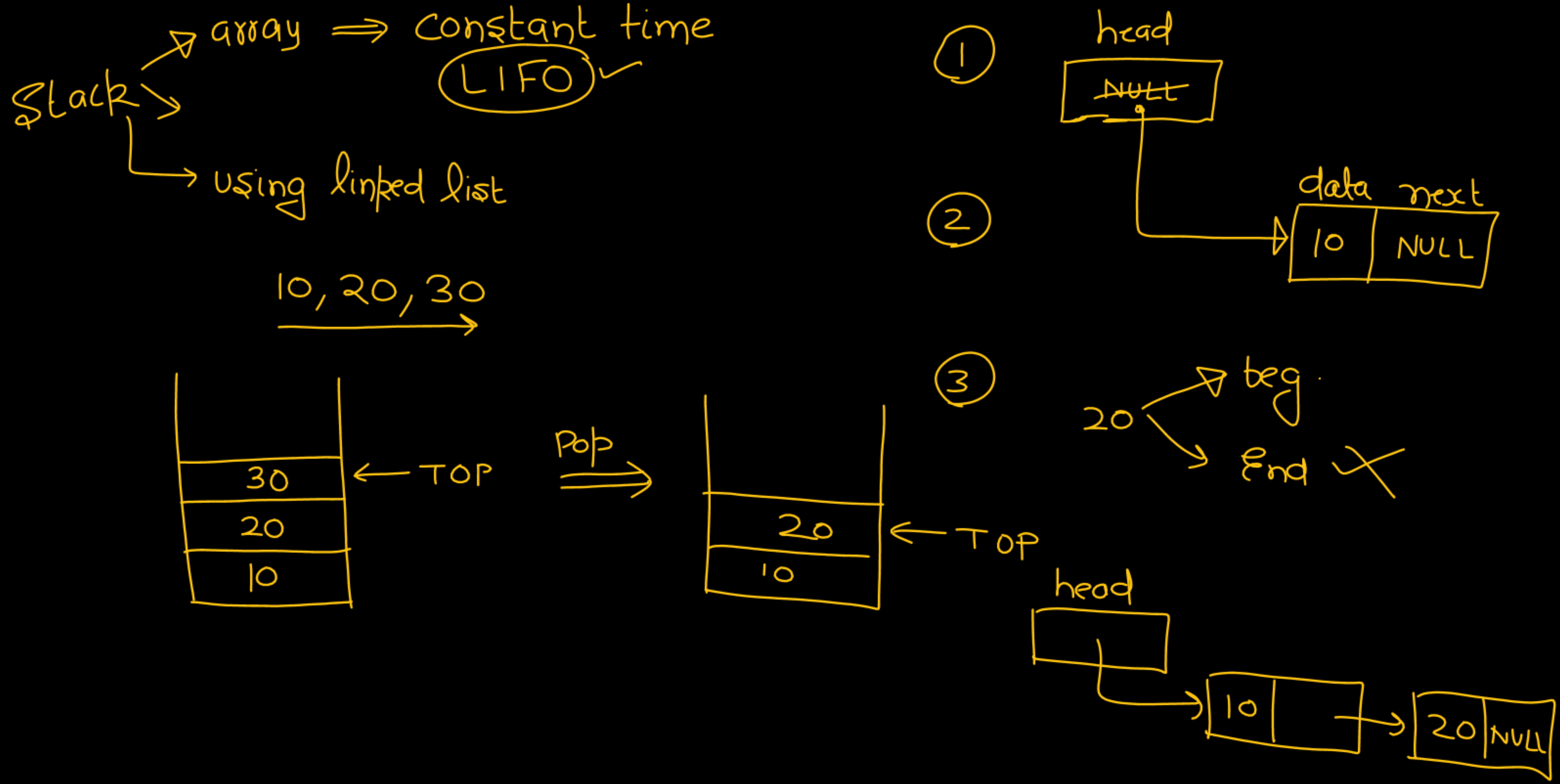
←

$+\ 2\ \times\ 3\ 5$

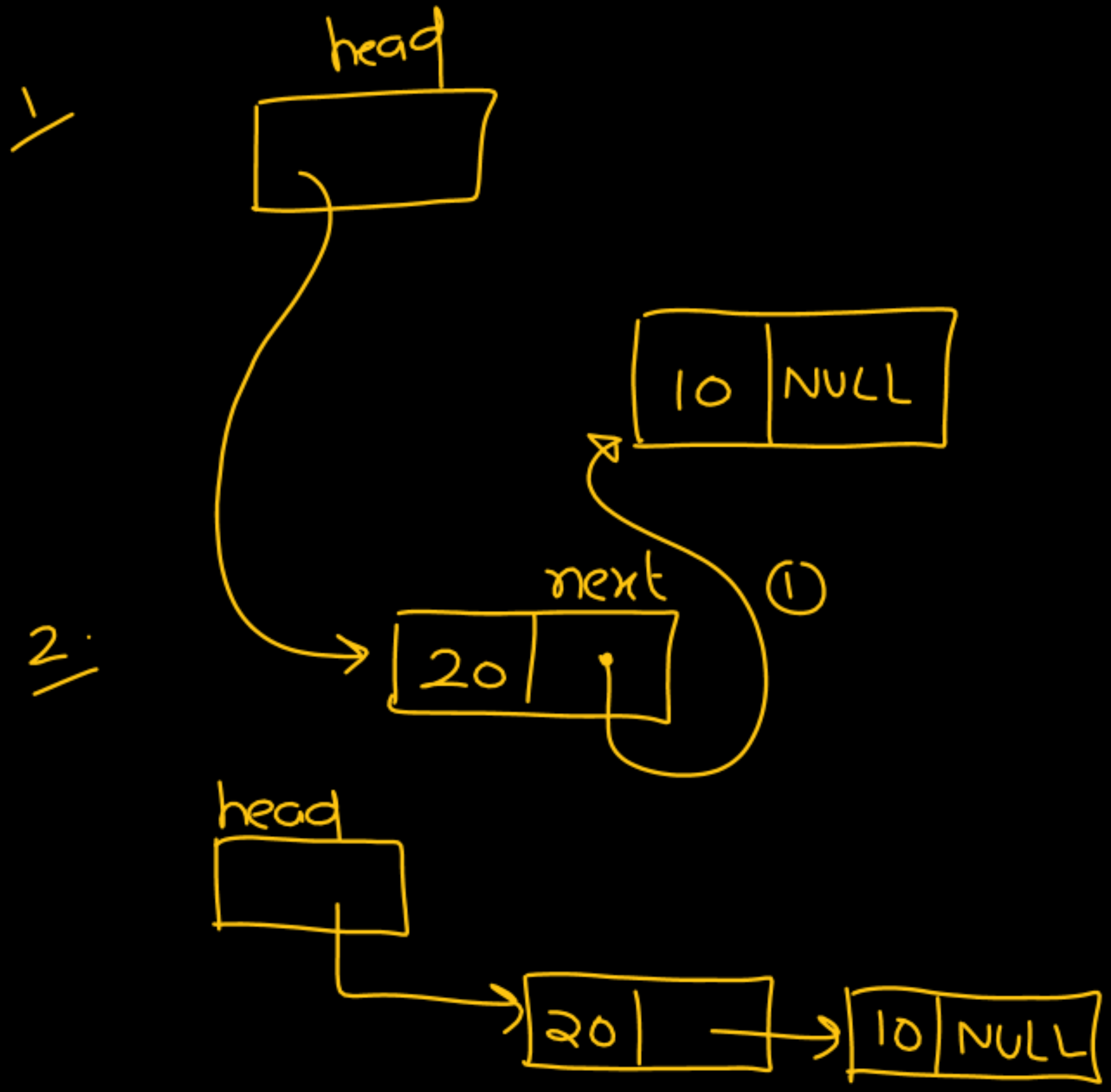
$+\ 2\ 15$

17

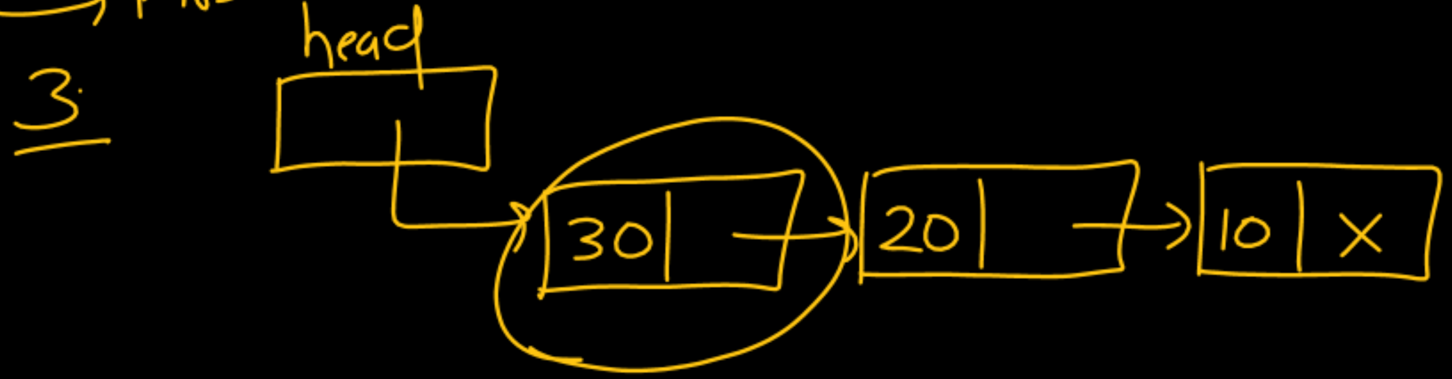
$\boxed{\text{Recursion}} \rightarrow$


Stack
array ⟹ constant time
LIFO ✓
using linked list
10, 20, 30
30
20
10
← TOP
Pop
20
10
← TOP
1
head
~~NULL~~
2
data next
10 NULL
3
20
beg
End ✗
head
10
20 NULL

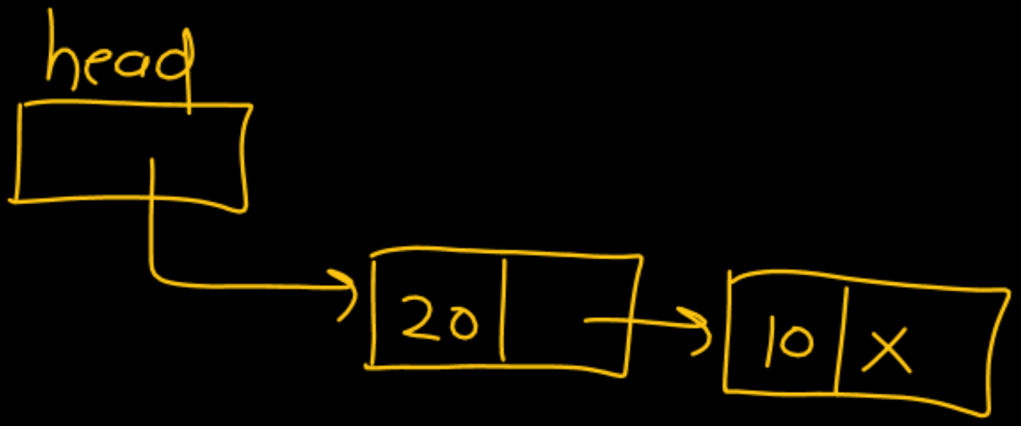
10



10, 20, 30 ← Last-in
→ First-out



Pop



A function f defined on stacks of integers satisfies the following properties.

f(∅) = 0 and f(push (S, i)) = max (f(S), 0) + i for all stacks S and integer i.

If a stack S contains the integers 2, - 3, 2, - 1, 2 in order from bottom to top, what is f(S)?

(A) 6

(B) 4

[✓ (C)] 3

(D) 2

$f(\text{Stack is Empty}) = 0$

(i) $f(\text{Push}(S,2)) = \max(0,0) + 2 = \boxed{2}$

(ii) $f(\text{Push}(S,-3)) = \max(2,0) + (-3) = \boxed{-1}$

(iii) $f(\text{Push}(S,2)) = \max(-1,0) + 2 = \boxed{2}$

(iv) $f(\text{Push}(S,-1)) = \max(2,0) + -1 = 1$

(v) $f(\text{Push}(S,2)) = \max(1,0) + 2 = \boxed{3}$

✓	2
✓	-1
✓	2
✓	-3
✓	2

The result of evaluating the postfix expression
10 5 + 60 6/ * 8 – is

A 284

B 213

[✓ C] 142

D 71

10 5 + 60 6 / × 8 −

15 60 6 / × 8 −

15 10 × 8 −

150 8 −

142

The best data structure to check whether an arithmetic expression has balanced parentheses is-

- (A) queue
- (B) stack
- (C) tree
- (D) list

~~# LP = # RP~~

) ( ) (

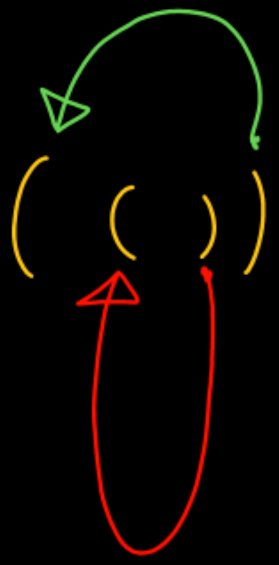
$(a+b) \times (a \times e)$

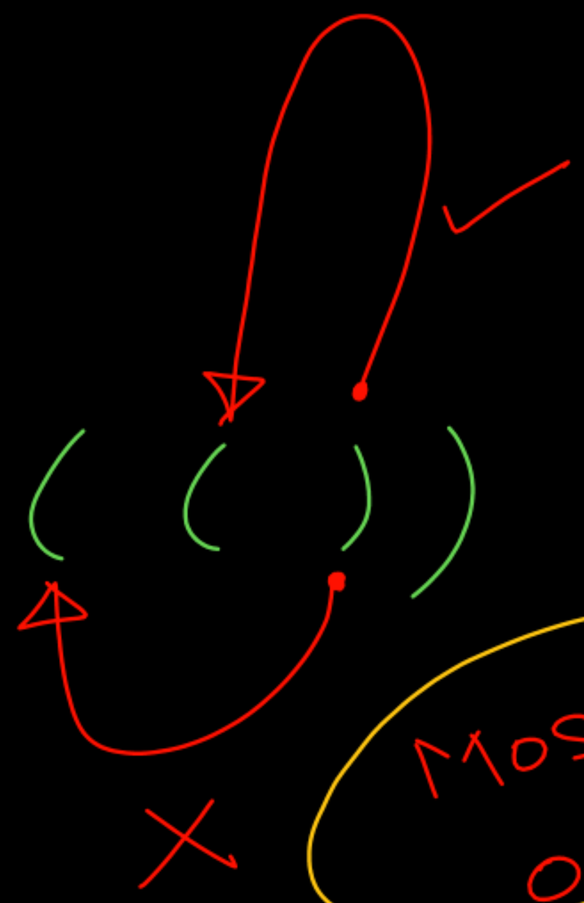
$\Downarrow$

( ) ( )

$((a+b) \times (a \times e))$

(()())





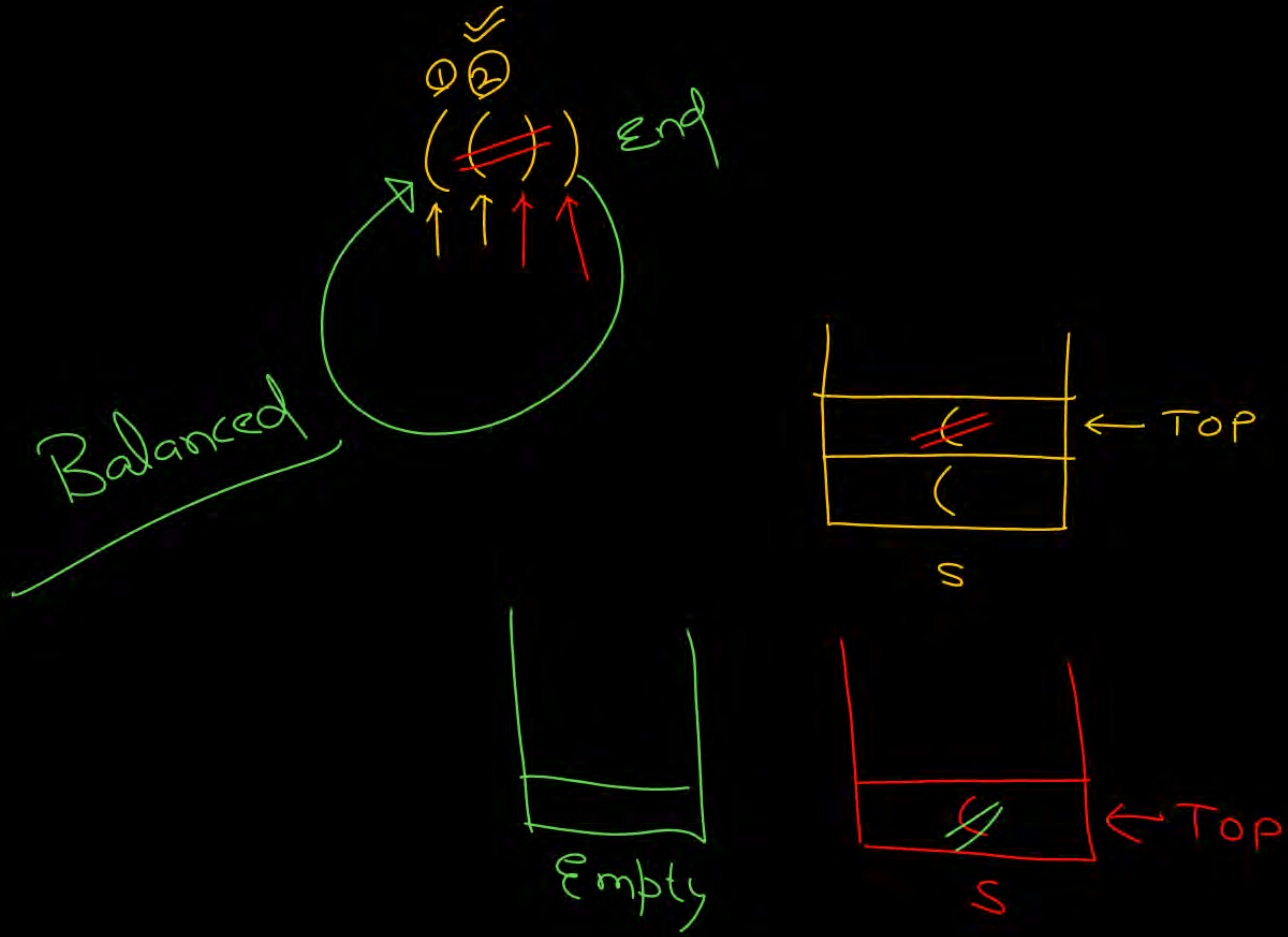
right par.

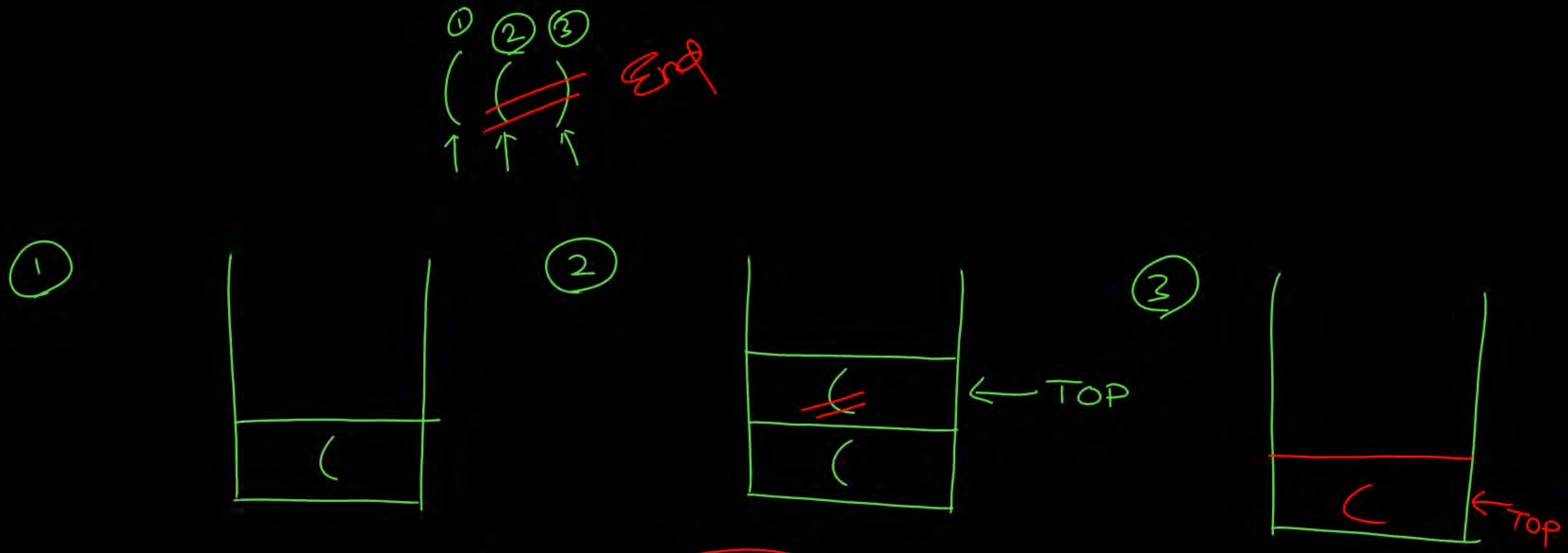
⇓

Most recently occur.

left paranthesis

TOP ⇒


✓
1 2
(( ))
End
Balanced
TOP
S
Empty
Top
S



End Empty

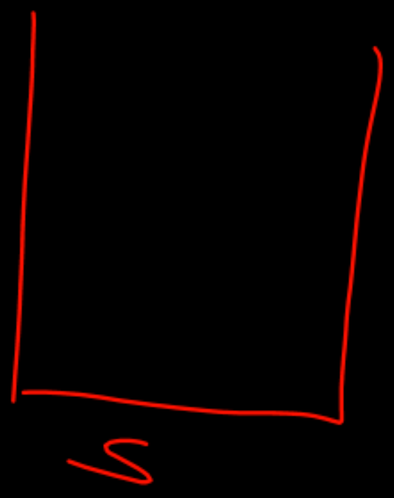
Unbalanced

) ( )

↑ → X

stack ⇒ Empty

Unbalanced



Which of the following is essential to convert an infix expression to post-fix expression efficiently?

[✓ A] An operator stack

B An operand stack

C An operator and an operand stack

D A parse tree

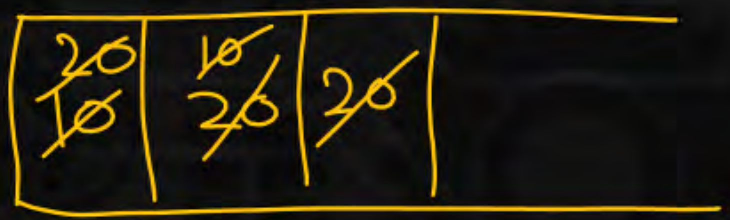
The following sequence of operations is performed on stack: PUSH(10), PUSH(20), POP, PUSH(10), PUSH(20), POP, POP, POP, PUSH(20), POP. The sequence of the value popped out is-

A 20 10 20 10 20

[✓ B] 20 20 10 10 20

C 10 20 20 10 20

D 20 20 10 20 10



20 20 10 1020

Which of the following permutations can be obtained in the output (in the same order) using a stack assuming that the input sequence is 1, 2, 3, 4, 5 in that order?

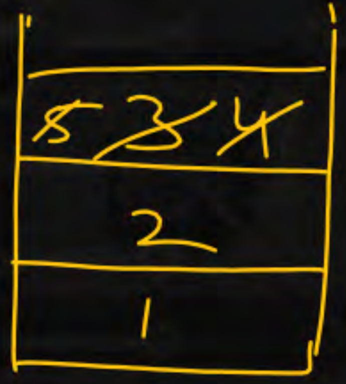
✓ ✓ ✓ ✗

~~(A)~~ 3, 4, 5, 1, 2

[✓ (B)] 3, 4, 5, 2, 1

(C) 1, 5, 2, 3, 4

(D) 5, 4, 3, 1, 2



A program attempts to generate as many permutations as possible of the string "abcd" by pushing the character a, b, c, d in the same order onto a stack but it may pop off the top character at any time. Which one of the following strings CANNOT be generated using this program?

~~A~~ abcd

~~B~~ dcba

~~C~~ cbad

[✓ D] cabd

Stack Permutation

The postfix expression corresponding to the infix expression $a+b*c-d\wedge\underbrace{e\wedge f}$ is-

$$a+b\times c-\underbrace{d\wedge[ef\wedge]}$$

$$a+\underbrace{b\times c}-[def\wedge\wedge]$$

$$\underbrace{a+[bc\times]}-[def\wedge\wedge]$$

$$[abc\times+]-[def\wedge\wedge]$$

$$abc\times+def\wedge\wedge-$$

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

8 2 3 ^ / 2 3 * + 5 1 * -

Note that ^ is the exponentiation operator. The top two elements of the stack after the first * is evaluated are:

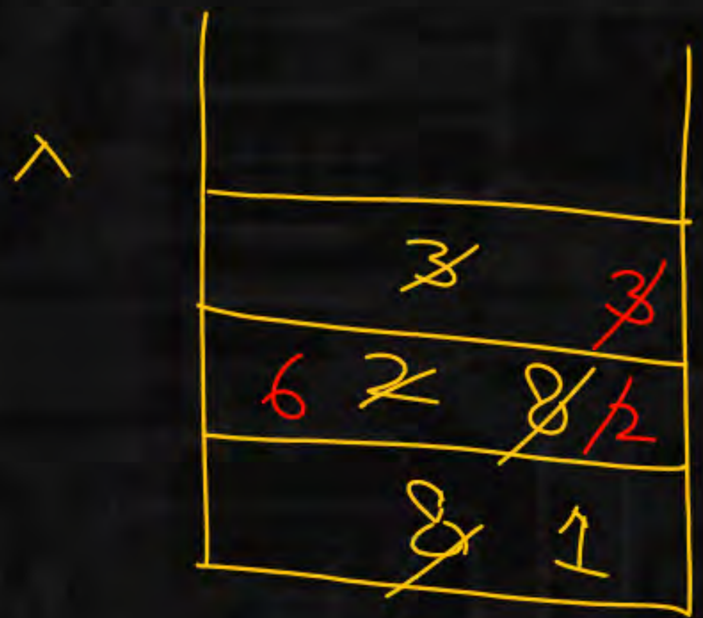
A. 6, 1

B. 5, 7

C. 3, 2

D. 1, 5

8 2 3 ^ / 2 3 × + 5 1 × −



3 2

$2^3 = 8$

×

3 2

$2 \times 3 = \boxed{6}$

Consider the following C program:

```
#include <stdio.h>
#define EOF -1
void push (int);
/*push the argument on the stack*/
int pop (void); /*pop the top of the stack*/
void flagError();
int main()
{
        int c, m, n, r,
        while((c = getchar())! = EOF){
        if (isdigit(c) push(c);
        else if ((c = = '+') || (c = = '*')){
            m = pop();
            n = pop();
            r = (c = = '+') ? n + m : n*m;
            push(r);
         }else if (c ! = '')
          flagError( );
        }
        printf(" %c ", pop());
}
```

What is the output of the program for the following?

5 2* 3 3 2 + * +

C (last lecture)

A 15

[✓ B] 25

C 30

D 150

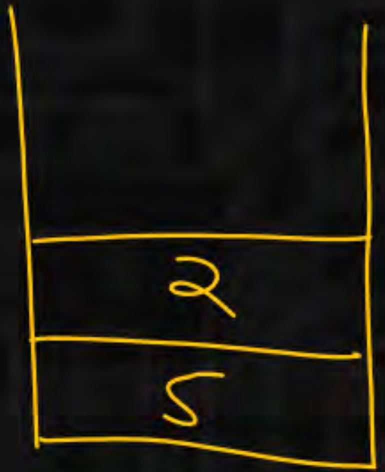
5 2 × 3 3 2 + × +

10 3 3 2 + × +

10 3 5 × +

10 15 +

⇒ 25



Let S be a stack of size n>=1. Starting with the empty stack, suppose we push the first n natural numbers in sequence, and then perform n pop operations. Assume that PUSH and POP operations take X secs each and Y seconds elapse between the end of one such stack operation and the start of the next operation. For m>=1, define the stack life-time of m as the time elapsed from the end f PUSH(m) to the start of POP operation that removes m from S. The average stack-life of an element is-

H.W

A n(X+Y)

B n(X+Y)-X

C 3Y+2X

D Y+2X

The attributes of three arithmetic operators in some programming language are given below.

Operator	Precedence	Associativity	Arity
+	High ✓	Left	Binary
– ✓	Medium	Right	Binary
*	Low	Left	Binary

×

$ Left

Right

@ left

+

The value of the expression 2 – 5 + 1– 7 * 3 in this language is__9____.

$(2 - ((5 + 1) - 7)) \times 3$

$(2 - (6 - 7)) \times 3$

$(2 - (-1)) \times 3$

3×3

$\boxed{9}$

THANK YOU GW SOLDIERS !