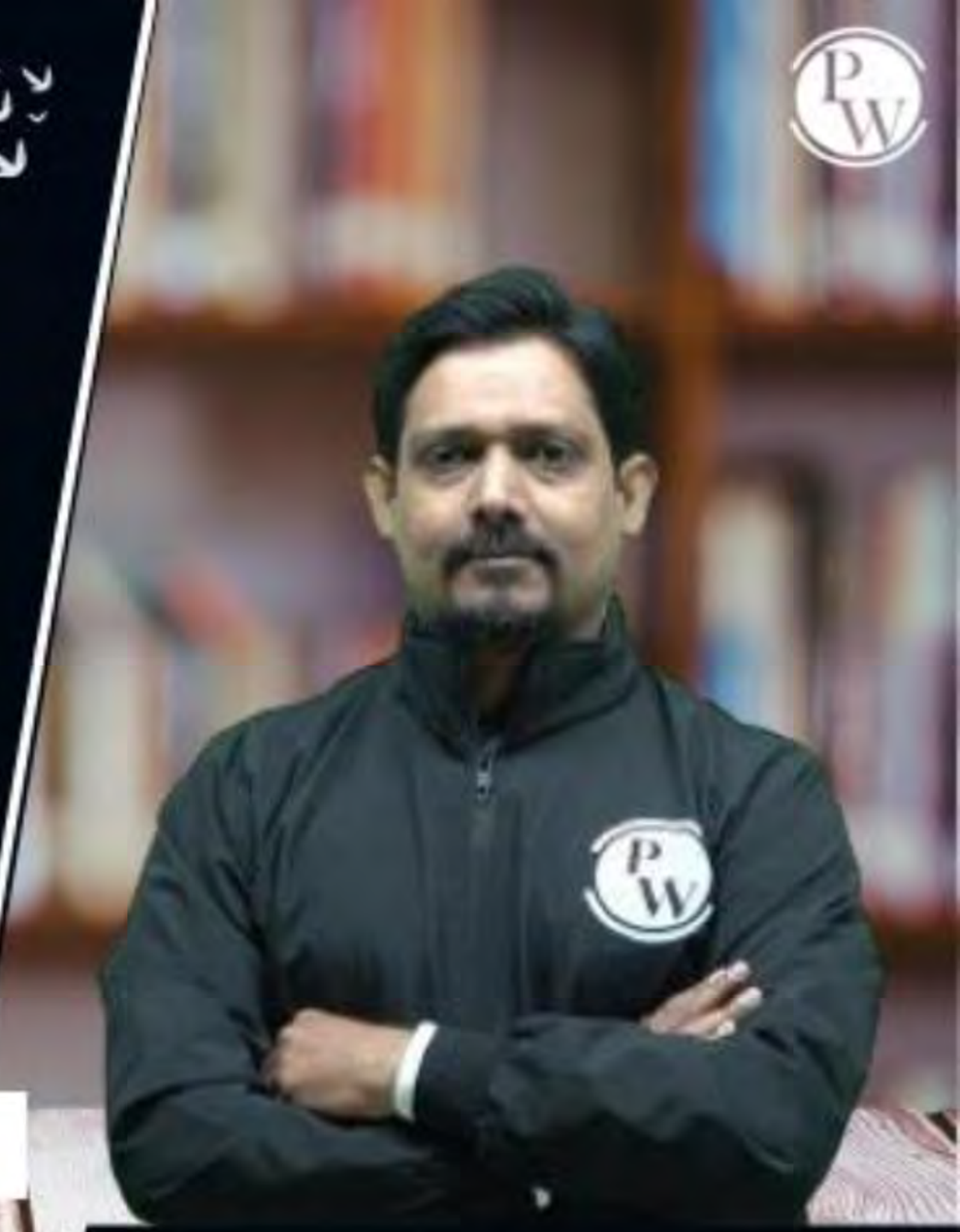
Computer Science & IT

Data Structure & Programming

Stack

Lecture No. 03



By- Abhishek Sir

Recap of Previous Lecture

Topic	Stack as Permutation Generator
Topic	Expression In computer
Topic	Practice Problem
Topic	
Topic	

Slide

Topics to be Covered

- Topic
- Topic
- Topic
- Topic
- Topic

Postfix & prefix using stack

Why postfix (Reverse polish), prefix (polish Notation)

Evaluation of postfix & prefix

Slide

Topic : Expression In Computer

A/B - C ↑ D ↑ E

3 4 2 1

postfix

= AB/ - C↑ DE↑

= AB/ - CDE↑↑

AB/ CDE↑↑ -

postfix

/AB - C↑ ↑DE

= /AB - ↑C↑DE

= -/AB ↑C↑DE

Slide

Topic : Expression In Computer

The **prefix** expression for the infix expression

a = – b +c * d/e + f ↑ g ↑ h – i * j

(a) = a – + + – b/*cde ↑f ↑gh*ij

(b) = a – + + – /*bcde ↑f ↑gh*ij

(c) = a + – + – b/*cde ↑f ↑gh*ij

(d) = a – + – – b/*cde ↑f ↑ah*ii

Perfection

Slide

The **prefix** expression for the infix expression

a = −b + c * d / e + f ↑ g ↑ h − i * j

(a) = a − + + − b/*cde ↑f ↑gh*ij

(b) = a − + + − /*bcde ↑f ↑gh*ij

(c) = a + − + − b/*cde ↑f ↑gh*ij

(d) = a − + − − b/*cde ↑f ↑ah*ii

unary $\boxed{-b}$

No chang in prefix care

−b

= a = -b + */cd/e + f ↑ ↑gh − *ij

= a = -b + /*cde + ↑f ↑gh − *ij

= a = +-b/*cde + ↑f ↑gh − *ij

= a = ++-b/*cde ↑f ↑gh − *ij

= a = −++-b/*cde ↑f ↑gh * ij

$\boxed{= a - ++ - b/*cde \uparrow f \uparrow gh * ij}$

Topic : Expression In Computer

Stack Rule : postfix

1 Scan one input at a time left to Right

2. if input is operand print the operand (left to right)

3. if input is operator & stack is empty push the operator

4 End of input pop everything from stack & print.

Slide

Topic : Expression In Computer

5. Input is operator & having more precedence then operator on top of stack then push

(HPU)

6 Input is operator & having lower precedence than operator on top of stack then pop operators

as long as input operator having lower precedence & push New operator.

LPoPu-s

Slide

Topic : Expression In Computer

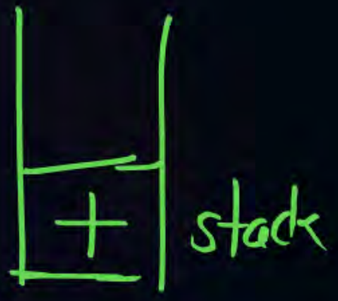
$a+bc*$

abc*+

$a+b*c$ → operator stack

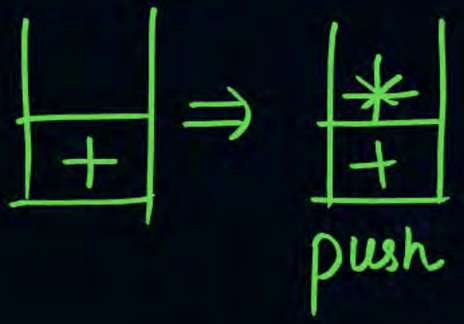
(I) Input : a
output : a

(II) Input : +
output : a



(III) Input : b
output : ab

(IV) Input : *
output : ab



(V) Input : c
output : abc

(VI) End of Input
abc*+

Slide

Topic : Expression In Computer

$a * b - c \Rightarrow \boxed{ab*} - c$

$\Rightarrow \boxed{ab*c-}$

(I) Input: a

output: a

(II) Input *

output: a

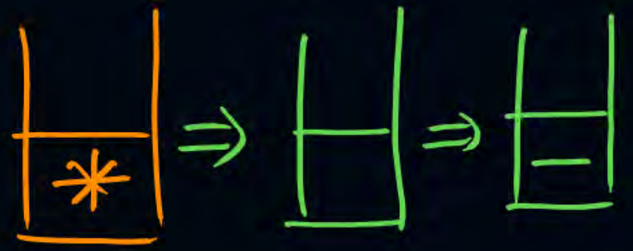
(III) Input: b

output: ab

(IV) Input: −

output: ab

ab*

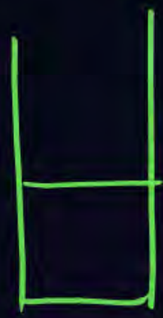


(V) Input: c

output: ab*c

(VI) . End of input

output: ab*c−



Empty Stack

Slide

Topic : Expression In Computer

$a * b \uparrow c - d$

(I) Input : a
output : a

(II) Input : *
output : a

(III) Input b
output : ab

(IV) Input : ↑
output : ab

(IV) Input : c
output abc

LPoPu-S

(VI) Input : −
output : abc↑*

(VII) Input : d
output : abc↑*d

(VIII) End of input
abc↑*d−

Slide

Topic : Expression In Computer

$a * b \uparrow c - d$

$= a * bc\uparrow - d$

$= abc\uparrow * d -$

$= abc\uparrow * d -$

Slide

Topic : Expression In Computer

(1) $(a +_1 b) +_2 c$ (Left Associative) $+_1$ $+_2$

$+_2$ is having lower precedence than $+_1$ LPoPu

$-$ S ← operators

} Associativity

$=_1$, $=_2$

(2) $a =_1 b =_2 c$ Right associative $=_2$ is higher precedence

HPU

Slide

Topic : Expression In Computer

prefix using stack

different

1. Scan Input Right to Left
2. print output Right to Left

7. LPoPu-S

Same

3 if input operator & stack is empty push the operator (same)

4 if input is operand then print (same)

5. End of input, pop Everything & print (Same)

6 HPu ← Same

Slide

Topic : Expression In Computer

Associativity precedence

Consider the operator precedence and associativity rules for the *integer* arithmetic operators given in the table below.

Operator	Precedence	Associativity
+	Highest	Left
-	High	Right
*	Medium	Right
/	Low	Right

The value of the expression $3 + 1 + 5 * 2/7 + 2 - 4 - 7 - 6/2$ as per the above rules is _____6____.

$3+1+5*2/7+2-4-7-6/2$

$= 9*2/(9-(4-(7-6)))/2$

$= 9*2/9-4-1/2$

$= 9*2/9-3/2$

$= 9*2/6/2 = 18/6/2 = 18/3$

$= 6$

Slide

Topic : Question

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

OPERATOR	PRECEDENCE	ASSOCIATIVITY	ARITY
+	High	Left	Binary
−	Medium	Right	Binary
*	Low	Left	Binary

The value of the expression $2 - 5 + 1 - 7 * 3$ in this language is 9.

$2-5+1-7*3$

$=2-6-7*3$

$=2-(6-7)*3$

$=2-(-1)*3$

$=3*3=9$

Slide

Topic : Question

#Q. The following postfix expression with single digit operands is evaluated using a stack:

823^ / 23 * + 51 * −

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

Slide

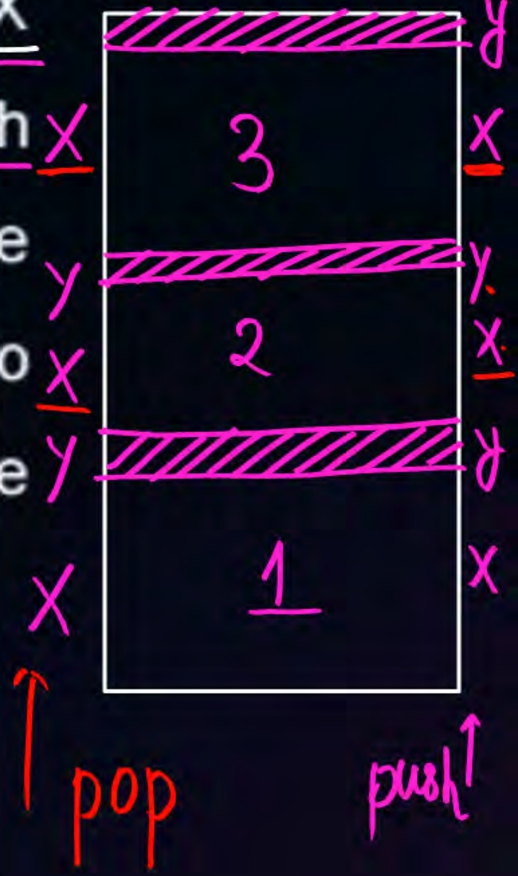
Topic : Question

HW

Maximum question y to be solved

Let s be a stack of size $n \geq 1$. 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 seconds each, and Y seconds elapse between the end of one such stack operation and the start of the next operation. For $m \geq 1$, define the stack-life of m as the time elapsed from the end of Push (m) to the start of the pop operation that removes m form S. The average stack life of an element of this stack is

3 stack Life y.



(A) $n(X + Y)$

(B) $3Y + 2X$

(C) $n(X + Y) - X$

(D) $Y + 2X$

2 mins Summary

- Topic
- Topic
- Topic
- Topic
- Topic

using stack poefix & postfix

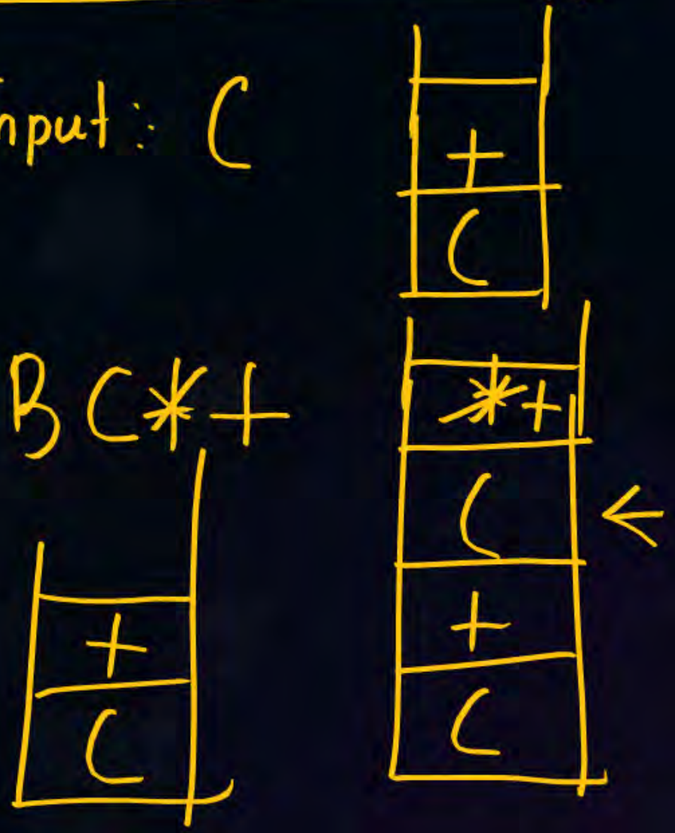
poachice pooblem

Stack Life

$(A+(B*C+D)/E)$

Input : C

ABC*+



Slide

THANK - YOU

Slide