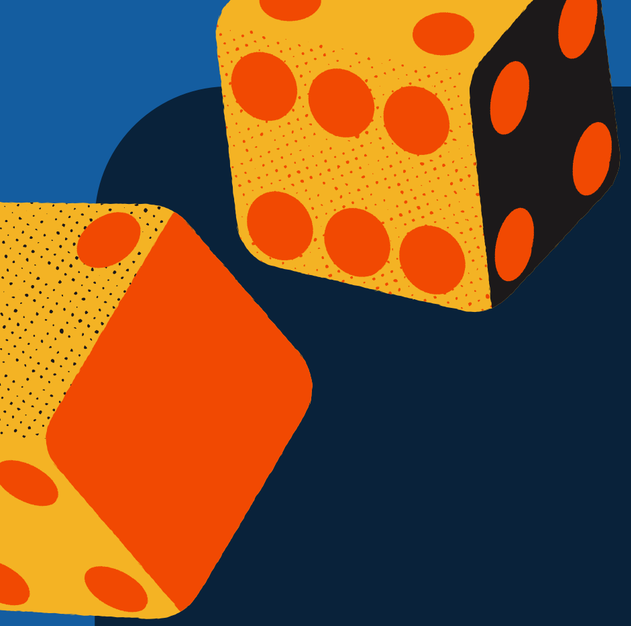
Welcome to Stack NooN!

Welcome to Stack NOON!

TechnoCrats GSV

BONUS !

Are you ready?



QUESTIONS:

- Minimum Add to Make Parentheses Valid
- String Manipulation
- Parenthesis Checker
- Smaller on Left
- Make the array beautiful
- Next Greater Element
- Next Greater Element in Circular Array
- Print Bracket Number
- Background String comparison

DEFINE :

A stack is a linear data structure which uses the some principle

LIFO (Last-In, First-Out) data structure

Implemented : an array or a linked list.

IMPLEMENTATION :

Array

Linked list

Array

Memory Efficient

Easier to implement and understand

L-list

NO Fixed size required

https://www.w3schools.com/dsa/dsa_data_stacks.php

2


1 2 3
6 5 4
7 8 9
12 11 10
14 15
18 17 16
19 20 21
24 23 22
26 27
30 29
31 32 33
36 35 34
37 38 39
41 40
43 44 45

3

OPERTAIONS :

Peek

Returns the top element on the stack

Push

Adds a new element on the stack.

Overflow

Pop

Removes and returns the top element from the stack

Underflow

https://www.w3schools.com/dsa/dsa_data_stacks.php

OPERTAIONS :

Overflow

Underflow

isEmpty

Checks if the stack is empty

Size

Finds the number of elements in the stack.

https://www.w3schools.com/dsa/dsa_data_stacks.php

OPERTAIONS ALGO :

Peek

```
Step 1: IF TOP = NULL
            PRINT "STACK IS EMPTY"
            Goto Step 3
Step 2: RETURN STACK[TOP]
Step 3: END
```

Push

```
Step 1: IF TOP = MAX-1
            PRINT "OVERFLOW"
            Goto Step 4
        [END OF IF]
Step 2: SET TOP = TOP + 1
Step 3: SET STACK[TOP] = VALUE
Step 4: END
```

Pop

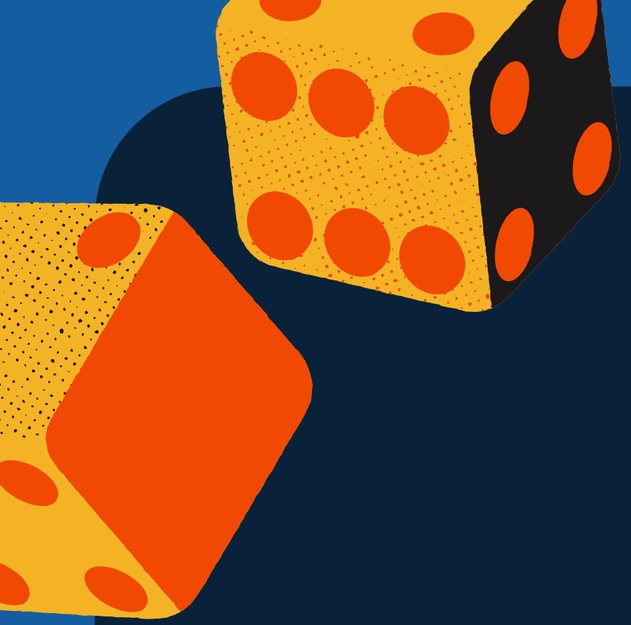
```
Step 1: IF TOP = NULL
            PRINT "UNDERFLOW"
            Goto Step 4
        [END OF IF]
Step 2: SET VAL = STACK[TOP]
Step 3: SET TOP = TOP - 1
Step 4: END
```

https://www.w3schools.com/dsa/dsa_data_stacks.php

BONUS !

Are you ready?

O(1)



1

2

Pop Culture

WHEN DOES “UNDERFLOW” OCCUR?

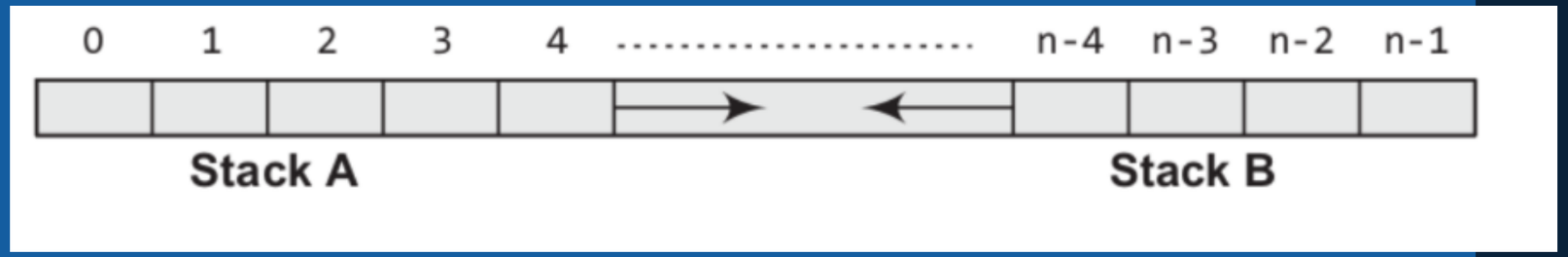
ANSWER NO. 1

Try deleting from an empty stack

MULTIPLE STACKS :

Trade off:

- OverFlow
- UnderFlow



APPLICATION :

Reversing

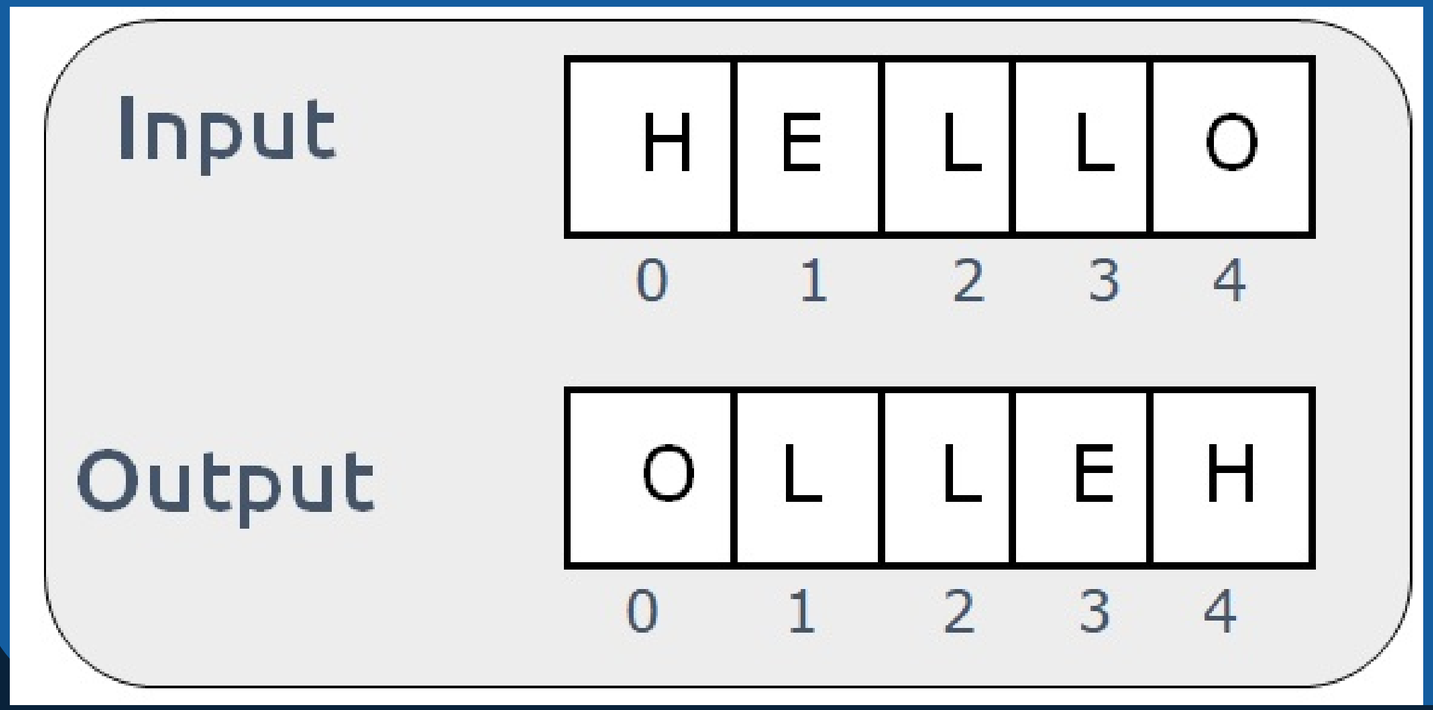
Parentheses checker

Infix Prefix Postfix

Recursion

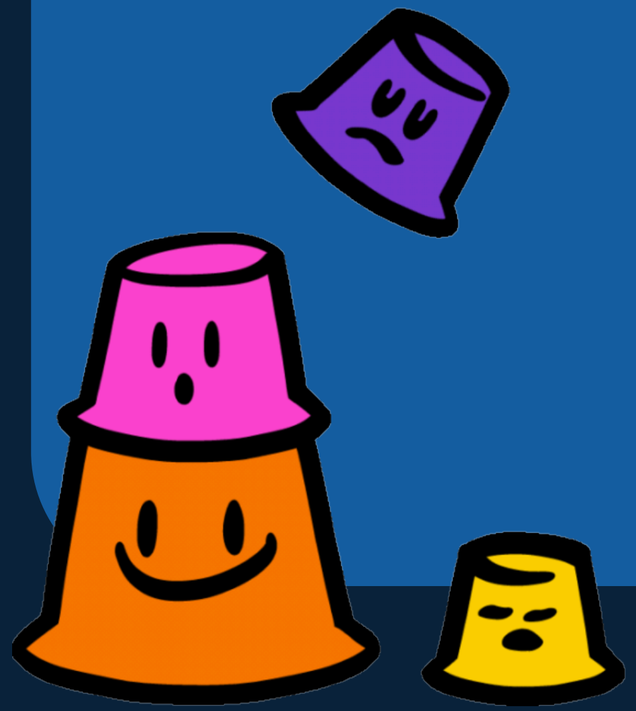
APPLICATION :

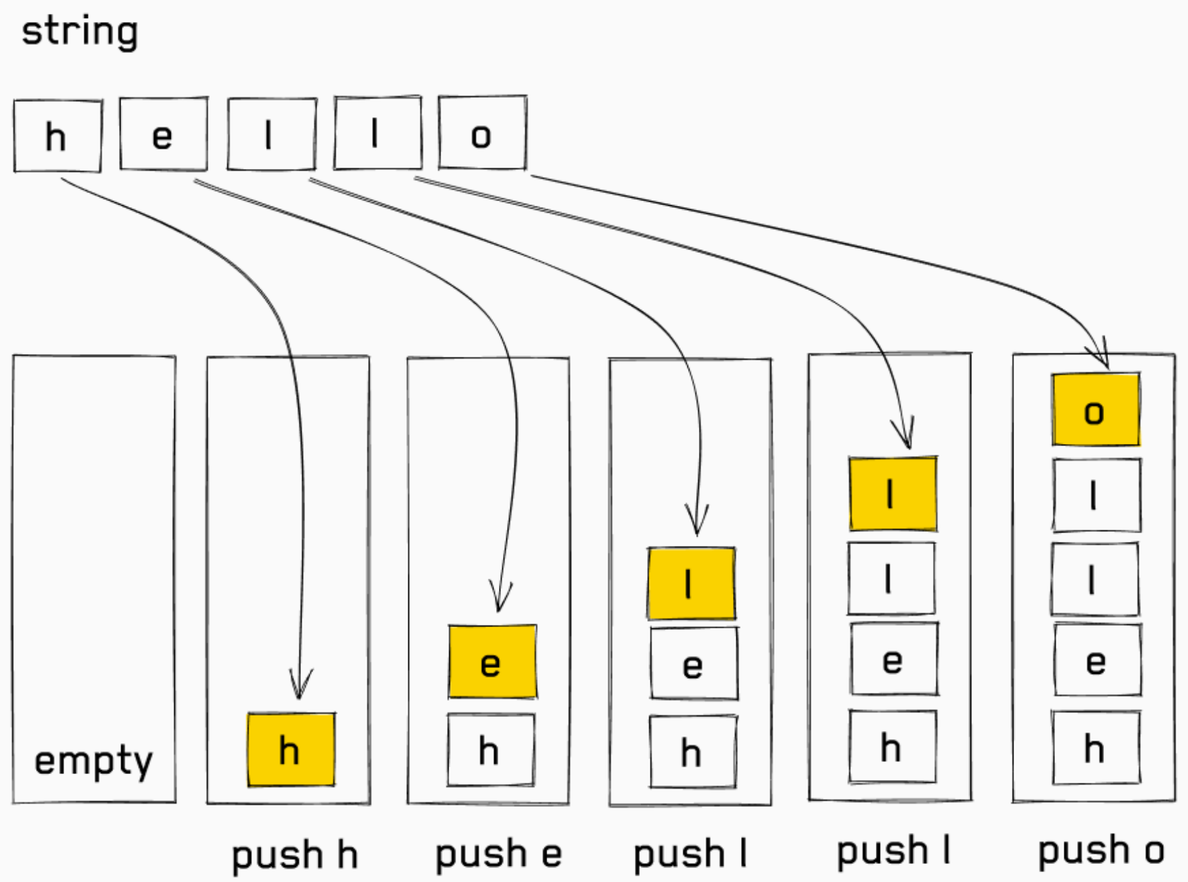
Reversing



APPLICATION :

Reversing





interviewing.io

PARENTHESES CHECKER :

(A+B}

{A+ (B – C)}

(A + B) / (C + D) – (D * E)

Algebraic expression contain parentheses, operands, and operators.

PARENTHESES CHECKER :

(A+B}

{A+ (B – C)}

(A + B) / (C + D) – (D * E)

Idea :

PUSH :

(

or { or

[

POP:

)

or } or

]

Visuals parachecker GFG

APPLICATION :

Notation		
	INFIX	A+(B*C+D)/E
	POSTFIX	A B C * D + E / +
	PREFIX	+ A / + * B C D E

INFIX TO POSTFIX :

(A + B) * C

(A–B) * (C+D)

(A + B) / (C + D)
–
(D * E)

INFIX TO POSTFIX :

(A + B) * C

- (AB+)*C
- AB+C*

(A–B) * (C+D)

[AB-] * [CD+]

- AB-CD+*

(A + B) / (C + D) – (D * E)

- [AB+] / [CD+] – [DE*]
- [AB+CD+/] – [DE*]
- AB+CD+/DE*-

INFIX TO POSTFIX :

(A + B) * C

- (+AB)*C
- *+ABC

(A–B) * (C+D)

[–AB] * [+CD]

- *–AB+CD

(A + B) / (C + D) – (D * E)

- [+AB] / [+CD] – [*DE]
- [/+AB+CD] – [*DE]
- –/+AB+CD*DE

INFIX TO POSTFIX :

Infix Scanned Character	Stack	Postfix Expressi
	(	
A	(	A
+	(+	A
(	(+ (	A
B	(+ (	A B
*	(+ (*	A B
C	(+ (*	A B C
+	(+ (+	A B C *
D	(+ (+	A B C * D
)	(+	A B C * D +
/	(+ /	A B C * D +
E	(+ /	A B C * D + E
		A B C * D + E / +

EVALUATION :

Postfix

Infix: 9 – ((3 * 4) + 8) / 4

Postfix: 9 3 4 * 8 + 4 / –

Character Scanned	Stack
9	9
3	9, 3
4	9, 3, 4
*	9, 12
8	9, 12, 8
+	9, 20
4	9, 20, 4
/	9, 5
–	4

EVALUATION :

Prefix

Prefix expression: + - 2 7 * 8 / 4 12

Character scanned	Operand stack
12	12
4	12, 4
/	3
8	3, 8
*	24
7	24, 7
2	24, 7, 2
-	24, 5
+	29

INFIX TO POSTFIX :

```
Step 1: Add ")" to the end of the infix expression
Step 2: Push "(" on to the stack
Step 3: Repeat until each character in the infix notation is scanned
        IF a "(" is encountered, push it on the stack
        IF an operand (whether a digit or a character) is encountered, add it to the
        postfix expression.
        IF a ")" is encountered, then
          a. Repeatedly pop from stack and add it to the postfix expression until a
             "(" is encountered.
          b. Discard the "(". That is, remove the "(" from stack and do not
             add it to the postfix expression
        IF an operator X is encountered, then
          a. Repeatedly pop from stack and add each operator (popped from the stack) to the
             postfix expression which has the same precedence or a higher precedence than X
          b. Push the operator X to the stack
        [END OF IF]
Step 4: Repeatedly pop from the stack and add it to the postfix expression until the stack is empty
Step 5: EXIT
```

INFIX TO POSTFIX :

```
Step 1: Add a ")" at the end of the
        postfix expression
Step 2: Scan every character of the
        postfix expression and repeat
        Steps 3 and 4 until ")"is encountered
Step 3: IF an operand is encountered,
        push it on the stack
        IF an operator O is encountered, then
        a. Pop the top two elements from the
           stack as A and B as A and B
        b. Evaluate B O A, where A is the
           topmost element and B
           is the element below A.
        c. Push the result of evaluation
           on the stack
        [END OF IF]
Step 4: SET RESULT equal to the topmost element
        of the stack
Step 5: EXIT
```

Converting an expression from Infix to postfix

(without parentheses)

INFIX TO PREFIX :

Step 1: Scan each character in the infix expression. For this, repeat Steps 2-8 until the end of infix expression
Step 2: Push the operator into the operator stack, operand into the operand stack, and ignore all the left parentheses until a right parenthesis is encountered
Step 3: Pop operand 2 from operand stack
Step 4: Pop operand 1 from operand stack
Step 5: Pop operator from operator stack
Step 6: Concatenate operator and operand 1
Step 7: Concatenate result with operand 2
Step 8: Push result into the operand stack
Step 9: END

INFIX TO PREFIX :

Step 1: Reverse the infix string. Note that while reversing the string you must interchange left and right parentheses.
Step 2: Obtain the postfix expression of the infix expression obtained in Step 1.
Step 3: Reverse the postfix expression to get the prefix expression

4

6

THANK YOU FOR PARTICIPATING

We hope you had fun!