PRACTICE PROGRAMMING ASSISGNMENTS **NOT GRADED**
PROF. NAVARTI SAXENA
DATA STRUCTURES AND ALGORITHMS

Stack

1. Implement "Push" and "Pop" in Stack

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

- push(x) -- Push element x onto stack.
- pop() -- Removes the element on top of the stack.
- top() -- Get the top element.
- getMin() -- Retrieve the minimum element in the stack.

Example 1:

Input
["MinStack","push","push","push","getMin","pop","top","getMin"]

[[],[-2],[0],[-3],[],[],[],[]]

Output
[null,null,null,null,-3,null,0,-2]

Explanation

```
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin(); // return -3
minStack.pop();
minStack.top();    // return 0
minStack.getMin(); // return -2
```

https://leetcode.com/explore/learn/card/queue-stack/230/usage-stack/1360/

2. Valid Parenthesis Checking

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

- Open brackets must be closed by the same type of brackets.
- Open brackets must be closed in the correct order.

Example 1:

```
Input: s = "()"
Output: true
```

Example 2:

```
Input: s = "()[]{}"
Output: true
```

Example 3:

```
Input: s = "(]"
Output: false
```

Example 4:

```
Input: s = "([)]"
Output: false
```

Example 5:

```
Input: s = "{[]}"
Output: true
```

Constraints:

- 1 <= s.length <= 104
- s consists of parentheses only '()[]{}'.

https://leetcode.com/explore/learn/card/queue-stack/230/usage-stack/1361/

3. Evaluate Reverse Polish Notation

Evaluate the value of an arithmetic expression in Reverse Polish Notation (Postfix).

Valid operators are +, -, *, /. Each operand may be an integer or another expression.

Note:

- Division between two integers should truncate toward zero.
- The given RPN expression is always valid. That means the expression would always evaluate to a result and there won't be any divide by zero operation.

- **Example 1:**

```
Input: ["2", "1", "+", "3", "*"]
Output: 9
Explanation: ((2 + 1) * 3) = 9
```

- **Example 2:**

```
Input: ["4", "13", "5", "/", "+"]
Output: 6
Explanation: (4 + (13 / 5)) = 6
```

- **Example 3:**

```
Input: ["10", "6", "9", "3", "+", "-11", "*", "/", "*", "17", "+", "5", "+"]
Output: 22
Explanation:
  ((10 * (6 / ((9 + 3) * -11))) + 17) + 5
= ((10 * (6 / (12 * -11))) + 17) + 5
= ((10 * (6 / -132)) + 17) + 5
= ((10 * 0) + 17) + 5
= (0 + 17) + 5
= 17 + 5
= 22
```

https://leetcode.com/explore/learn/card/queue-stack/230/usage-stack/1394/