Deadline : 7th September 2021, 8:59pm

1. Create separate class(java file) for each problem
2. Test your solution with few examples in main method
3. Try to avoid writing logic in the main method, create separate method for the logic to execute
4. Create different folder for problems for each assignment
5. Try to include edge cases in your logic

Q1) Create a Shape class and two sub classes Rectangle and Square

Shape :

Fields : name, color, area, perimeter

Constructors :
Shape(name, color)
Shape(name, color, perimeter, area)

Methods : printShape()

Rectangle: Inherits Shape

Fields : width, height

Constructors:
Rectangle(side) → If single side is given then set width and height to same value.
Rectangle(width, height)
Rectangle(name, color, width, height)

Square: Inherits Shape

Fields: side

Constructors:
Square(side)
Square(name, color, side)

1) Create getter and setter methods for all classes.
2) Create printShape() method in Shape class that returns desired string output.
3) Override the getArea() and getPerimeter() getter methods of Shape in Rectangle and Square to give desired output.

4)Use the Shape tester class to test your classes.

```java
public class ShapeTester {
  public static void main(String[] args) {
    Shape shape1 = new Shape("Shape1", "pink");
    System.out.println("shape1");
    System.out.println(shape1.printShape());

    Shape shape2 = new Shape("Shape2", "orange", 20, 30);
    System.out.println("shape2");
    System.out.println("area : " + shape2.getArea() + " perimeter : " + shape2.getPerimeter());
    System.out.println(shape2.printShape());

    Rectangle rectangle1 = new Rectangle(2);
    System.out.println("rectangle1");
    System.out.println("area : " + rectangle1.getArea() + " perimeter : " + rectangle1.getPerimeter());

    Rectangle rectangle2 = new Rectangle("Rectangle", "Purple",4, 7);
    System.out.println("rectangle2");
    System.out.println("area : " + rectangle2.getArea() + " perimeter : " + rectangle2.getPerimeter());
    System.out.println(rectangle2.printShape());

    Square square1 = new Square(3);
    System.out.println("square1");
    System.out.println("area : " + square1.getArea() + " perimeter : " + square1.getPerimeter());

    Square square2 = new Square("Square" , "black", 7);
    System.out.println("square2");
    System.out.println("area : " + square2.getArea() + " perimeter : " + square2.getPerimeter());
    System.out.println(square2.printShape());
  }
}
```

Output:

```
shape1
The Shape1 has a pink color
shape2
area : 30 perimeter : 20
The Shape2 has a orange color
rectangle1
area : 4 perimeter : 8
rectangle2
area : 28 perimeter : 22
The Rectangle has a Purple color
square1
area : 9 perimeter : 12
square2
area : 49 perimeter : 28
The Square has a black color
```

Q2) Create a Sum class with methods to get desired output. Use the below function to test the output.

```
public static void main(String[] args){
    Sum sum = new Sum();
    System.out.println(sum.add(2, 3)); //output : 5
    System.out.println(sum.add(4, 9, 12));//output : 25
    System.out.println(sum.add(4, 5.0));//output : 9.0
    System.out.println(sum.add(15.5, 5));//output : 20.5
    System.out.println(sum.add(1.0, 6.4));//output : 7.4
}
```

LeetCode :

1) **Transpose Matrix** : https://leetcode.com/problems/transpose-matrix/

Given a 2D integer array `matrix`, return *the* ***transpose*** *of* `matrix`.

The **transpose** of a matrix is the matrix flipped over its main diagonal, switching the matrix's row and column indices.

Example 1:

```
Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]

Output: [[1,4,7],[2,5,8],[3,6,9]]
```

Example 2:

```
Input: matrix = [[1,2,3],[4,5,6]]

Output: [[1,4],[2,5],[3,6]]
```

Constraints:

- `m == matrix.length`
- `n == matrix[i].length`
- `1 <= m, n <= 1000`
- $1 <= m * n <= 10^5$
- $-10^9 <= matrix[i][j] <= 10^9$

2) **Shortest Word Distance :** https://leetcode.com/problems/sign-of-the-product-of-an-array/

Given an array of strings `wordsDict` and two different strings that already exist in the array `word1` and `word2`, return *the shortest distance between these two words in the list*.

Example 1:

```
Input: wordsDict = ["practice", "makes", "perfect", "coding", "makes"], word1 =
"coding", word2 = "practice"

Output: 3
```

Example 2:

```
Input: wordsDict = ["practice", "makes", "perfect", "coding", "makes"], word1 =
"makes", word2 = "coding"

Output: 1
```

Constraints:

- `1 <= wordsDict.length <= 3 * 10`^{4}
- `1 <= wordsDict[i].length <= 10`
- `wordsDict[i]` consists of lowercase English letters.
- `word1` and `word2` are in `wordsDict`.
- `word1 != word2`

3) **Move Zeroes** : https://leetcode.com/problems/move-zeroes/

Given an integer array `nums`, move all `0`'s to the end of it while maintaining the relative order of the non-zero elements.

Note that you must do this in-place without making a copy of the array.

Example 1:

```
Input: nums = [0,1,0,3,12]

Output: [1,3,12,0,0]
```

Example 2:

```
Input: nums = [0]

Output: [0]
```

Constraints:

- `1 <= nums.length <=` 10^4
- -2^{31} `<= nums[i] <=` $2^{31} - 1$

4) Rotate Image - https://leetcode.com/problems/rotate-image/

You are given an n x n 2D matrix representing an image, rotate the image by **90** degrees (clockwise).
You have to rotate the image **in-place**, which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

Example 1:

```
Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]

Output: [[7,4,1],[8,5,2],[9,6,3]]
```

Example 2:

```
Input: matrix = [[5,1,9,11],[2,4,8,10],[13,3,6,7],[15,14,12,16]]

Output: [[15,13,2,5],[14,3,4,1],[12,6,8,9],[16,7,10,11]]
```

Example 3:

```
Input: matrix = [[1]]

Output: [[1]]
```

Example 4:

```
Input: matrix = [[1,2],[3,4]]

Output: [[3,1],[4,2]]
```

Constraints:

- `matrix.length == n`
- `matrix[i].length == n`
- `1 <= n <= 20`
- `-1000 <= matrix[i][j] <= 1000`

5) **Spiral Matrix** - https://leetcode.com/problems/spiral-matrix/

Given an `m x n` `matrix`, return *all elements of the* `matrix` *in spiral order*.

Example 1:

```
Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]

Output: [1,2,3,6,9,8,7,4,5]
```

Example 2:

```
Input: matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]

Output: [1,2,3,4,8,12,11,10,9,5,6,7]
```

Constraints:

- `m == matrix.length`
- `n == matrix[i].length`
- `1 <= m, n <= 10`
- `-100 <= matrix[i][j] <= 100`

6) **Isomorphic Strings** - https://leetcode.com/problems/isomorphic-strings/

Given two strings `s` and `t`, *determine if they are isomorphic*.

Two strings `s` and `t` are isomorphic if the characters in `s` can be replaced to get `t`.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.

Example 1:

```
Input: s = "egg", t = "add"

Output: true
```

Example 2:

```
Input: s = "foo", t = "bar"

Output: false
```

Example 3:

```
Input: s = "paper", t = "title"

Output: true
```

Constraints:

- `1 <= s.length <= 5 * 10`^{4}
- `t.length == s.length`
- `s` and `t` consist of any valid ascii character.

7) **Add Strings** - https://leetcode.com/problems/add-strings/

Given two non-negative integers, `num1` and `num2` represented as string, return *the sum of* `num1` *and* `num2` *as a string*.

You must solve the problem without using any built-in library for handling large integers (such as `BigInteger`). You must also not convert the inputs to integers directly.

Example 1:

```
Input: num1 = "11", num2 = "123"

Output: "134"
```

Example 2:

```
Input: num1 = "456", num2 = "77"

Output: "533"
```

Example 3:

```
Input: num1 = "0", num2 = "0"

Output: "0"
```

Constraints:

- `1 <= num1.length, num2.length <= 10`^{4}
- `num1` and `num2` consist of only digits.
- `num1` and `num2` don't have any leading zeros except for the zero itself.

8) **Valid Palindrome** - https://leetcode.com/problems/valid-palindrome/

Given a string `s`, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.

Example 1:

```
Input: s = "A man, a plan, a canal: Panama"

Output: true

Explanation: "amanaplanacanalpanama" is a palindrome.
```

Example 2:

```
Input: s = "race a car"

Output: false

Explanation: "raceacar" is not a palindrome.
```

Constraints:

- `1 <= s.length <= 2 * 10`^{5}
- `s` consists only of printable ASCII characters.

9) **Reverse words in a String** - https://leetcode.com/problems/reverse-words-in-a-string/

Given an input string `s`, reverse the order of the **words**.

A **word** is defined as a sequence of non-space characters. The **words** in `s` will be separated by at least one space.

Return *a string of the words in reverse order concatenated by a single space.*

Note that `s` may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces.

Example 1:

```
Input: s = "the sky is blue"

Output: "blue is sky the"
```

Example 2:

```
Input: s = "  hello world  "

Output: "world hello"

Explanation: Your reversed string should not contain leading or trailing spaces.
```

Example 3:

```
Input: s = "a good   example"

Output: "example good a"

Explanation: You need to reduce multiple spaces between two words to a single space
in the reversed string.
```

Example 4:

```
Input: s = "  Bob    Loves  Alice   "

Output: "Alice Loves Bob"
```

Example 5:

```
Input: s = "Alice does not even like bob"

Output: "bob like even not does Alice"
```

Constraints:

- `1 <= s.length <= 10`^{4}
- `s` contains English letters (upper-case and lower-case), digits, and spaces `' '`.
- There is **at least one** word in `s`.

10) String Compression - https://leetcode.com/problems/string-compression/

Given an array of characters `chars`, compress it using the following algorithm:

Begin with an empty string `s`. For each group of **consecutive repeating characters** in `chars`:

- If the group's length is `1`, append the character to `s`.
- Otherwise, append the character followed by the group's length.

The compressed string `s` **should not be returned separately**, but instead, be stored **in the input character array `chars`**. Note that group lengths that are `10` or longer will be split into multiple characters in `chars`.

After you are done **modifying the input array**, return *the new length of the array*.

You must write an algorithm that uses only constant extra space.

Example 1:

```
Input: chars = ["a","a","b","b","c","c","c"]
```

```
**Output:** Return 6, and the first 6 characters of the input array should be:
["a","2","b","2","c","3"]

**Explanation:** The groups are "aa", "bb", and "ccc". This compresses to "a2b2c3".
```

Example 2:

```
Input: chars = ["a"]

Output: Return 1, and the first character of the input array should be: ["a"]

Explanation: The only group is "a", which remains uncompressed since it's a single
character.
```

Example 3:

```
Input: chars = ["a","b","b","b","b","b","b","b","b","b","b","b","b"]

Output: Return 4, and the first 4 characters of the input array should be:
["a","b","1","2"].

Explanation: The groups are "a" and "bbbbbbbbbbbb". This compresses to "ab12".
```

Example 4:

```
Input: chars = ["a","a","a","b","b","a","a"]

Output: Return 6, and the first 6 characters of the input array should be:
["a","3","b","2","a","2"].

Explanation: The groups are "aaa", "bb", and "aa". This compresses to "a3b2a2". Note
that each group is independent even if two groups have the same character.
```

Constraints:

- `1 <= chars.length <= 2000`
- `chars[i]` is a lowercase English letter, uppercase English letter, digit, or symbol.