35. Problem: 1389. Create Target Array in the Given Order.

Given two arrays of integers `nums` and `index`. Your task is to create *target* array under the following rules:
Initially *target* array is empty.
From left to right read nums[i] and index[i], insert at index `index[i]` the value `nums[i]` in *target* array.
Repeat the previous step until there are no elements to read in `nums` and `index`.
Return the *target* array.
It is guaranteed that the insertion operations will be valid.

```
Example 1:
Input: nums = [0,1,2,3,4], index = [0,1,2,2,1]
Output: [0,4,1,3,2]
Explanation:
nums       index     target
0            0         [0]
1            1         [0,1]
2            2         [0,1,2]
3            2         [0,1,3,2]
4            1         [0,4,1,3,2]
Example 2:
Input: nums = [1,2,3,4,0], index = [0,1,2,3,0]
Output: [0,1,2,3,4]
Explanation:
nums       index     target
1            0         [1]
2            1         [1,2]
3            2         [1,2,3]
4            3         [1,2,3,4]
0            0         [0,1,2,3,4]
Example 3:
Input: nums = [1], index = [0]
Output: [1]
```

```
Constraints:
1 <= nums.length, index.length <= 100
nums.length == index.length
0 <= nums[i] <= 100
0 <= index[i] <= i
```

Solution:

My Solution:

```java
class Solution_1389 {
    public int[] createTargetArray(int[] nums, int[] index) {
        List<Integer> list = new ArrayList<>();
        for (int i = 0; i < index.length; i++) {
            list.add(index[i], nums[i]);
        }
        int[] array = new int[list.size()];
```

```java
        for (int i = 0; i < list.size(); i++) {
            array[i] = list.get(i);
        }
        return array;
    }
}
```

// https://leetcode.com/problems/create-target-array-in-the-given-order/

Another Solution: 01

```java
class Solution {
    public int[] createTargetArray(int[] nums, int[] index) {
        int[] target = new int[nums.length];
        int i = 0, k = 0;
        while (i < index.length) {
            for (k = target.length - 1; k > index[i]; k--)
                target[k] = target[k - 1];
            target[index[i]] = nums[i];
            i++;
        }
        return target;
    }
}
```

Another Solution: 02

```java
class Solution {
    public int[] createTargetArray(int[] nums, int[] index) {
        ArrayList<Integer> al = new ArrayList<>();
        for(int i=0 ;i<nums.length;i++){
            al.add(index[i],nums[i]);
        }
        int[] arr = new int[al.size()];
        for(int i=0 ;i<nums.length;i++)
            arr[i]=al.get(i);

        return arr;
    }
}
```

Another Solution: 03

```java
class Solution {
    public int[] createTargetArray(int[] nums, int[] index) {
        int[] target = new int[nums.length];
        for(int i = 0; i < index.length; i++){
            int ind = index[i];
            if(ind < i){
                int j = i;
                while(j > ind){
                    target[j] = target[j - 1];
```

```
                target[j - 1] = nums[i];
                j--;
            }
        }
        else{
            target[i] = nums[ind];
        }
    }
    return target;
    }
}
```

34. Problem: 2006. Count Number of Pairs with absolute Difference K.

Given an integer array `nums` and an integer `k`, return *the number of pairs* `(i, j)` *where* `i < j` *such that* `|nums[i] - nums[j]| == k`.
The value of `|x|` is defined as:
x if `x >= 0`.
-x if `x < 0`.

Example 1:
```
Input: nums = [1,2,2,1], k = 1
Output: 4
Explanation: The pairs with an absolute difference of 1 are:
- [1,2,2,1]
- [1,2,2,1]
- [1,2,2,1]
- [1,2,2,1]
```
Example 2:
```
Input: nums = [1,3], k = 3
Output: 0
Explanation: There are no pairs with an absolute difference of 3.
```
Example 3:
```
Input: nums = [3,2,1,5,4], k = 2
Output: 3
Explanation: The pairs with an absolute difference of 2 are:
- [3,2,1,5,4]
- [3,2,1,5,4]
- [3,2,1,5,4]
```

Constraints:
```
1 <= nums.length <= 200
1 <= nums[i] <= 100
1 <= k <= 99
```

Solution:

My Solution:

```
class Solution_2006 {
    public int countKDifference(int[] nums, int k) {
        int c = 0;
```

```
    for (int i = 0; i < nums.length; i++) {
        for (int j = i + 1; j < nums.length; j++) {
            if ((nums[j] - nums[i]) == (-k) || (nums[j] - nums[i]) == (k)) {
                c++;
            }
        }
    }
    return c;
    }
}
```

// https://leetcode.com/problems/count-number-of-pairs-with-absolute-difference-k/submissions/

Another Solution: 01

```
class Solution {
    public int countKDifference(int[] nums, int k) {
        Map<Integer, Integer> map = new HashMap<>();
        int res = 0;
        for (int i = 0; i < nums.length; i++) {
            if (map.containsKey(nums[i] - k)) {
                res += map.get(nums[i] - k);
            }
            if (map.containsKey(nums[i] + k)) {
                res += map.get(nums[i] + k);
            }
            map.put(nums[i], map.getOrDefault(nums[i], 0) + 1);
        }
        return res;
    }
}
```

Another Solution: 02

```
//Brute Force
class Solution {
    public int countKDifference(int[] nums, int k) {
        int count = 0;
        for (int i = 0; i < nums.length; i++)
            for (int j = i + 1; j < nums.length; j++)
                if (Math.abs(nums[j] - nums[i]) == k) count++;
        return count;
    }
}
```

Another Solution: 03

```
class Solution {
    public int countKDifference(int[] a, int k) {
        int count = 0;
        int fre[] = new int[101];
        for (int x : a) {
```

```
            if (x - k >= 0) count += fre[x - k];
            if (x + k <= 100) count += fre[x + k];
            fre[x]++;
        }
        return count;
    }
}
```

33. Problem: 2418. Sort the People

You are given an array of strings `names`, and an array `heights` that consists of distinct positive integers. Both arrays are of length n.

For each index i, `names[i]` and `heights[i]` denote the name and height of the ith person.

Return `names` *sorted in descending order by the people's heights*.

Example 1:
```
Input: names = ["Mary","John","Emma"], heights = [180,165,170]
Output: ["Mary","Emma","John"]
Explanation: Mary is the tallest, followed by Emma and John.
```
Example 2:
```
Input: names = ["Alice","Bob","Bob"], heights = [155,185,150]
Output: ["Bob","Alice","Bob"]
Explanation: The first Bob is the tallest, followed by Alice and the second Bob.
```

Constraints:
```
n == names.length == heights.length
1 <= n <= 10³
1 <= names[i].length <= 20
1 <= heights[i] <= 10⁵
```
`names[i]` consists of lower and upper case English letters.
All the values of `heights` are distinct.

Solution:

My Solution:

```java
class Solution_2418 {
    public String[] sortPeople(String[] names, int[] heights) {
        int n = heights.length;
        HashMap<Integer, String> hm = new HashMap<>();
        for (int i = 0; i < n; i++) {
            hm.put(heights[i], names[i]);
        }
        Arrays.sort(heights);
        for (int j = 0; j < n; j++) {
            names[n - j - 1] = hm.get(heights[j]);
        }
        return names;
    }
}
```

// https://leetcode.com/problems/sort-the-people/submissions/

Another Solution: 01

```java
class Solution {
    public String[] sortPeople(String[] names, int[] heights) {
        int n = heights.length;
        String temp;
        int t;
        for (int i = 0; i < n; i++) {
            for (int j = 1; j < (n - i); j++) {
                if (heights[j - 1] < heights[j]) {
                    t = heights[j - 1];
                    heights[j - 1] = heights[j];
                    heights[j] = t;

                    temp = names[j - 1];
                    names[j - 1] = names[j];
                    names[j] = temp;
                }
            }
        }
        return names;
    }
}
```

Another Solution: 02

```
/* we cannot sort input heights directly, because after sorted it we cannot know which height is belong to which
person.
So before sort, we need to add some extra information (in this case is "indices") to heights.
Just image sorting "heights" and every "height" has a name tag on them, using the name tag we can know it's
who's height.
This is the 2D array comes from.
    Then we can use build-in sort to sort the 2d array.
    After sorted, we can get name via indices. */
```

```java
//Runtime: 9 ms, faster than 80.00% of Java online submissions for Sort the People.
//Memory Usage: 42.9 MB, less than 100.00% of Java online submissions for Sort the People.
//build-in sort
//Time: O(NlgN + N); Space: O(N + lgN)
//Time: O(NlgN); Space: O(N)

class Solution {
    public String[] sortPeople(String[] names, int[] heights) {
        //Space: O(N)
        //form a 2D array using people's height and their position in original array.
        int[][] people = new int[names.length][2];
        for (int i = 0; i < names.length; i++)
            people[i] = new int[]{heights[i], i};
```

```java
        //Time: O(NlgN); Space: O(lgN)
        //sort in descending order by the people's heights.
        Arrays.sort(people, (a, b) -> b[0] - a[0]);

        String[] res = new String[names.length];
        //Time: O(N)
        //Get result from sorted array.
        for (int i = 0; i < names.length; i++)
            res[i] = names[people[i][1]];

        return res;
    }
}
```

Another Solution: 03

```
/* Intuition
People usually either uses quick sort or Hash map/ tree map to solve this problem. I found another way without
using Map to achieve that using 2 dimensional array
Approach
    First, we need to make a two dimensional array containing index and height
    Sort descendent by height
    Fill the result array with names according to index
    Code */
```

```java
class Solution {
    public String[] sortPeople(String[] names, int[] heights) {
        int arrLength = names.length;
        int[][] heightOfPeople = new int[arrLength][2];

        // Init array with current index and actual height
        for (int i = 0; i < arrLength; i++) {
            heightOfPeople[i][0] = i;
            heightOfPeople[i][1] = heights[i];
        }

        // Sort descendent by height
        Arrays.sort(heightOfPeople, (x, y) -> y[1] - x[1]);

        // Fill the result array with names according to index
        String[] results = new String[arrLength];
        for (int i = 0; i < arrLength; i++) {
            results[i] = names[heightOfPeople[i][0]];
        }

        return results;
    }
}
```

32. Problem: 804. Unique Morse Code Words

International Morse Code defines a standard encoding where each letter is mapped to a series of dots and dashes, as follows:
'a' maps to ".-",
'b' maps to "-...",
'c' maps to "-.-.", and so on.
For convenience, the full table for the 26 letters of the English alphabet is given below:
[".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-",".-..","--","-.","---",".--.","--.-",".-.","...","-","..-","...-",".--","-..-","-.--","--.."]
Given an array of strings words where each word can be written as a concatenation of the Morse code of each letter.
For example, "cab" can be written as "-.-..--...", which is the concatenation of "-.-.", ".-", and "-...". We will call such a concatenation the transformation of a word.
Return *the number of different transformations among all words we have*.

Example 1:
```
Input: words = ["gin","zen","gig","msg"]
Output: 2
Explanation: The transformation of each word is:
"gin" -> "--...-."
"zen" -> "--...-."
"gig" -> "--...--."
"msg" -> "--...--."
There are 2 different transformations: "--...-." and "--...--.".
```
Example 2:
```
Input: words = ["a"]
Output: 1
```

Constraints:
```
1 <= words.length <= 100
1 <= words[i].length <= 12
```
words[i] consists of lowercase English letters.

Solution:

My Solution:

```java
class Solution_804 {
    public int uniqueMorseRepresentations(String[] words) {
        String[] mappings = {".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.", "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--.."};
        Set<String> set = new HashSet();
        StringBuilder code = new StringBuilder();
        for (String s : words) {
            for (int i = 0; i < s.length(); i++) {
                code.append(mappings[s.charAt(i) - 'a']); // -'a' to get array index; Ascii val of a = 97
            }
            set.add(code.toString());
            code.setLength(0); //clearing the stringbuilder
        }
        return set.size();
    }
}
```

```
}
```

Another Solution: 01

```java
//Java Stream
class Solution {
    public int uniqueMorseRepresentations(String[] words) {
        String[] alphabet = new String[]{".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", ".---", "-.-", ".-.."
            ,"--", "-.", "---", ".--.", "--.-", ".-.", "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--.."};
        return (int) Arrays.stream(words).map(word -> word.chars().mapToObj(character -> alphabet[character - 'a'])
            .collect(Collectors.joining())).distinct().count();
    }
}
```

Another Solution: 02

```java
class Solution {
    public int uniqueMorseRepresentations(String[] words) {
        HashMap<Character, String> map = new HashMap<>();
        map.put('a', ".-");      map.put('b', "-...");     map.put('c', "-.-.");     map.put('d', "-..");
        map.put('e', ".");       map.put('f', "..-.");     map.put('g', "--.");      map.put('h', "....");
        map.put('i', "..");      map.put('j', ".---");     map.put('k', "-.-");      map.put('l', ".-..");
        map.put('m', "--");      map.put('n', "-.");       map.put('o', "---");      map.put('p', ".--.");
        map.put('q', "--.-");    map.put('r', ".-.");      map.put('s', "...");      map.put('t', "-");
        map.put('u', "..-");     map.put('v', "...-");     map.put('w', ".--");      map.put('x', "-..-");
        map.put('y', "-.--");    map.put('z', "--..");

        HashSet<String> set = new HashSet<>();
        for (int i = 0; i < words.length; i++) {
            String str = words[i];
            String str1 = "";
            for (int j = 0; j < str.length(); j++) {
                str1 = str1 + map.get(str.charAt(j));
            }
            set.add(str1);
        }
        return set.size();

    }
}
```

Another Solution: 03

```java
class Solution {
    public int uniqueMorseRepresentations(String[] words) {
        String[] arr = new String[]{".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..", ".---", "-.-", ".-..", "--", "-.", "---
", ".--.", "--.-", ".-.", "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--.."};
```

```java
        HashSet<String> set = new HashSet<>();
        for (String word : words)
            set.add(transformMorse(word, arr));
        return set.size();
    }

    private String transformMorse(String word, String[] arr) {
        StringBuilder build = new StringBuilder();
        for (char c : word.toCharArray())
            build.append(arr[(int) c - 97]);
        return build.toString();
    }
}
```

31. Problem: 1588. Sum of All Odd Length Subarrays

Given an array of positive integers `arr`, return *the sum of all possible odd-length subarrays of* `arr`.
A subarray is a contiguous subsequence of the array.

Example 1:
```
Input: arr = [1,4,2,5,3]
Output: 58
Explanation: The odd-length subarrays of arr and their sums are:
[1] = 1
[4] = 4
[2] = 2
[5] = 5
[3] = 3
[1,4,2] = 7
[4,2,5] = 11
[2,5,3] = 10
[1,4,2,5,3] = 15
If we add all these together we get 1 + 4 + 2 + 5 + 3 + 7 + 11 + 10 + 15 = 58
```
Example 2:
```
Input: arr = [1,2]
Output: 3
Explanation: There are only 2 subarrays of odd length, [1] and [2]. Their sum is 3.
```
Example 3:
```
Input: arr = [10,11,12]
Output: 66
```

Constraints:
```
1 <= arr.length <= 100
1 <= arr[i] <= 1000
```

Solution:

My Solution:

```java
class Solution_1588 {
    public int sumOddLengthSubarrays(int[] arr) {
```

```
        int sum = 0;
        for (int i = 0; i < arr.length; i++)
            sum += (arr[i] * ((((i + 1) * (arr.length - i)) + 1) / 2));
        return sum;
    }
}
```

// https://leetcode.com/problems/sum-of-all-odd-length-subarrays/

Another Solution: 01

```
/*  Input array = [1,4,2,5,3]

    Subarrays and Subsequence are two different things. Eg- [1,4,2],[4,2,5] but not [1,4,5].
    Subarrays contains continuous array elements, whereas Subsequence contains 2^N elements(They can skip
inbetween elements to form an array). Eg-[1,4,5]
    Let us define all the subarrays:
    length 1
    1
    4
    2
    5
    3
    length 2
    1,4
    4,2
    2,5
    5,3
    length 3
    1,4,2
    4,2,5
    2,5,3
    length 4
    1,4,2,5
    4,2,5,3
    length 5
    1,4,2,5,3
    Here we need to only consider odd lengths - (length 1, length 3, length 5)
    We can see a pattern:
    0 1 2 3 4 (index)
    1 4 2 5 3 (array elements)
    3 4 5 4 3 (number of times array elements occurs in length 1, length 3, length 5). Eg- 4 occurs one time in
length 1 + two times in length 3 + one time in length 5.
    3 16 10 20 9 (Total= array elements * num of times it occurs) = 3+16+10+20+9 = 58.

    Challenge is to calculate number of times an array element occurs:
    Let us consider array element 1:
    Consider index 0
    Find how many subarrays start with that index = 1([1])+1([1,4])+1([1,4,2])+1([1,4,2,5])+1([1,4,2,5,3]) = 5.
```

Find how many subarrays end with that index = 1[1] = 1.

Consider index 1
Find how many subarrays start with that index = 1([1,4])+1([1,4,2])+1([1,4,2,5])+1([1,4,2,5,3]) = 4.
Find how many subarrays end with that index = 1[1] = 1.

Pattern: start = n-i
End = i+1
Total number of times : start * end (Remember this is for both odd and even)
Odd = Total / 2 (Consider adding extra one to the odd) (3 4 5 4 3)
Result = array element * odd
Do it for all the array elements and corresponding odd numbers and add it to result. (3+16+10+20+9=58)
*/

```java
class Solution {
    public int sumOddLengthSubarrays(int[] arr) {
        int result = 0;
        int n = arr.length;
        for (int i = 0; i < n; i++) {
            int end = i + 1;
            int start = n - i;
            int total = start * end;
            int odd = total / 2;
            if (total % 2 == 1) {
                odd++;
            }
            result += odd * arr[i];
        }
        return result;
    }
}
```

Another Solution: 02

```java
//Method 1: T: O(n³) | S: O(1) | Sliding Window (Brute force)
class Solution {
    public int sumOddLengthSubarrays3(int[] arr) {
        int total = 0;
        for (int window = 1; window <= arr.length; window += 2)
            for (int i = 0; i + window <= arr.length; i++)
                for (int j = i; j < i + window; j++)
                    total += arr[j];
        return total;
    }
}
```

```java
// Method 2: T: O(n²) | S: O(1) | Sliding Window (optimised)
class Solution {
    public int sumOddLengthSubarrays(int[] arr) {
```

```
        int total = 0;
        for (int windowSize = 1; windowSize <= arr.length; windowSize += 2) {
            // sum up all the elements in the first window, W1
            int windowSum = 0;
            for (int i = 0; i < windowSize; i++)
                windowSum += arr[i];
            total += windowSum;
            // slide W1 one element at a time.
            // remove (subtract) the first element from the W1 and insert (add) the last element of the slided W1 into
the previous sum.
            for (int i = 1; i + windowSize <= arr.length; i++) {
                windowSum = windowSum - arr[i - 1] + arr[i + windowSize - 1];
                total += windowSum;
            }
        }
        return total;
    }
}
```

Another Solution: 03

```
/* Method 3: T: O(n) | S: O(1) | Counting element occurence
```

courtesy: https://leetcode.com/problems/sum-of-all-odd-length-subarrays/discuss/980310/Dew-It-or-O(n)-1-
pass-or-(Almost)-No-math
A brilliant solution-explanation combo, please upvote it.
I've slightly refactored the above solution. Please read the original solution to understand better.
IMO, this solution is one of those that if you already know it, then you know it, else its not easy to conjure this
in a live interview, unless you have acute observation skills. No interviewer should expect this solution.
Expectation would be that you come up with Method 1 then optimise it to Method 2. */

```
class Solution {
    public int sumOddLengthSubarrays(int[] arr) {
        int n = arr.length, sum = 0;
        int evenStops = (n & 1) == 0 ? n / 2 : n / 2 + 1;
        int evenStarts = 1;
        int oddStarts = 0;
        int oddStops = n / 2;
        for (int i = 0; i < n; ++i) {
            sum += ((oddStops * oddStarts) + (evenStops * evenStarts)) * arr[i];
            if ((i & 1) == 0) {
                oddStarts++;
                evenStops--;
            } else {
                evenStarts++;
                oddStops--;
            }
        }
    }
```

```
        return sum;
    }
}
```

Day 06 -2194, 2325, 1720, 66, 136.

30. Problem: 2194. Cells in a Range on a Excel Sheet

A cell `(r, c)` of an excel sheet is represented as a string `"<col><row>"` where:
`<col>` denotes the column number `c` of the cell. It is represented by alphabetical letters.
For example, the 1st column is denoted by `'A'`, the 2nd by `'B'`, the 3rd by `'C'`, and so on.
`<row>` is the row number `r` of the cell. The rth row is represented by the integer `r`.
You are given a string `s` in the format `"<col1><row1>:<col2><row2>"`, where `<col1>` represents the
column c1, `<row1>` represents the row r1, `<col2>` represents the column c2, and `<row2>` represents the row r2, such
that `r1 <= r2` and `c1 <= c2`.
Return *the list of cells* `(x, y)` *such that* `r1 <= x <= r2` *and* `c1 <= y <= c2`. The cells should be represented
as strings in the format mentioned above and be sorted in non-decreasing order first by columns and then by rows.

Example 1:


```
Input: s = "K1:L2"
Output: ["K1","K2","L1","L2"]
Explanation:
The above diagram shows the cells which should be present in the list.
The red arrows denote the order in which the cells should be presented.
Example 2:
```



```
Input: s = "A1:F1"
Output: ["A1","B1","C1","D1","E1","F1"]
Explanation:
The above diagram shows the cells which should be present in the list.
The red arrow denotes the order in which the cells should be presented.
```

Constraints:
```
s.length == 5
'A' <= s[0] <= s[3] <= 'Z'
'1' <= s[1] <= s[4] <= '9'
```
s consists of uppercase English letters, digits and `':'`.

Solution:

My Solution:

```java
class Solution_2194 {
    public List<String> cellsInRange(String s) {
        char c1 = s.charAt(0), c2 = s.charAt(3);
        char r1 = s.charAt(1), r2 = s.charAt(4);
        List<String> cells = new ArrayList<>();
        for (char c = c1; c <= c2; ++c) {
            for (char r = r1; r <= r2; ++r) {
                cells.add("" + c + r);
            }
        }
        return cells;
    }
}
```

// https://leetcode.com/problems/cells-in-a-range-on-an-excel-sheet/

Another Solution: 01

```java
class Solution {
    public List < String > cellsInRange(String s) {
        String[] str = s.split(":");
        List < String > res = new ArrayList < > ();
        int a = Character.getNumericValue(str[0].charAt(1));
        int b = Character.getNumericValue(str[1].charAt(1));
        int c1 = str[0].charAt(0) - 'A';
        int c2 = str[1].charAt(0) - 'A';

        for (int i = c1; i <= c2; i++) {
            StringBuilder sb = new StringBuilder();
            sb.append(Character.toString(i + 65));
            for (int j = a; j <= b; j++) {
                sb.append(j);

                res.add(sb.toString());

                sb = sb.deleteCharAt(sb.length() - 1);
            }
        }
        return res;
    }
}
```

Another Solution: 02

```java
class Solution {
    public List cellsInRange(String s) {
        String a=s.substring(1,2);
```

```java
        String b=s.substring(s.length()-1,s.length());

        List<String> list=new ArrayList<>();
        char c=s.charAt(0);
        char d=s.charAt(3);

        if(a.equals(b)){
            for(char m=c;m<=d;m++){
                // System.out.println(m);
                list.add(m+a);
            }
        }
        else{
            int x=Integer.parseInt(a);
            int y=Integer.parseInt(b);
            for(int z=x;z<=y;z++){
                for(char m=c;m<=d;m++){
                    String last=Integer.toString(z);
                    list.add(m+last);
                }
            }
        }
        Collections.sort(list);
        return list;
    }
}
```

Another Solution: 03

```java
class Solution {
    public List<String> cellsInRange(String s) {
        String alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        String number = "0123456789";
        ArrayList<String> list = new ArrayList<>();
        String str = new String();
        int colIndexStart = alpha.indexOf(Character.toString(s.charAt(0)));
        int colIndexEnd = alpha.indexOf(Character.toString(s.charAt(3)));
        int rowIndexStart = Integer.valueOf(s.charAt(1)-48);
        int rowIndexEnd = Integer.valueOf(s.charAt(4)-48);
        for(int i=colIndexStart; i<=colIndexEnd; i++){
            for(int j=rowIndexStart; j<=rowIndexEnd; j++){
                str += alpha.charAt(i);
                str += number.charAt(j);
                list.add(str);
                str = "";
            }
        }
        return list;
    }
}
```

29. Problem: 2325. Decode the Message

You are given the strings `key` and `message`, which represent a cipher key and a secret message, respectively. The steps to decode `message` are as follows:

Use the first appearance of all 26 lowercase English letters in `key` as the order of the substitution table.

Align the substitution table with the regular English alphabet.

Each letter in `message` is then substituted using the table.

Spaces ' ' are transformed to themselves.

For example, given `key = "happy boy"` (actual key would have at least one instance of each letter in the alphabet), we have the partial substitution table of (`'h' -> 'a','a' -> 'b','p' -> 'c','y' -> 'd','b' -> 'e','o' -> 'f'`).

Return *the decoded message*.

Example 1:

t	h	e	q	u	i	c	k	b	r	o	w	n	f	x	j	m	p	s	v	l	a	z	y	d	g
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z

```
Input: key = "the quick brown fox jumps over the lazy dog", message = "vkbs bs t suepuv"
Output: "this is a secret"
Explanation: The diagram above shows the substitution table.
It is obtained by taking the first appearance of each letter in "the quick brown fox jumps over
the lazy dog".
```

Example 2:

e	l	j	u	x	h	p	w	n	y	r	d	g	t	q	k	v	i	s	z	c	f	m	a	b	o
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z

```
Input: key = "eljuxhpwnyrdgtqkviszcfmabo", message = "zwx hnfx lqantp mnoeius ycgk vcnjrdb"
Output: "the five boxing wizards jump quickly"
Explanation: The diagram above shows the substitution table.
It is obtained by taking the first appearance of each letter in "eljuxhpwnyrdgtqkviszcfmabo".
```

Constraints:
```
26 <= key.length <= 2000
```
`key` consists of lowercase English letters and ' '.

`key` contains every letter in the English alphabet ('a' to 'z') at least once.
```
1 <= message.length <= 2000
```
`message` consists of lowercase English letters and ' '.

Solution:

My Solution:

```
class Solution_2325 {
    public String decodeMessage(String key, String message) {
        StringBuilder ans = new StringBuilder();//Using String Builder to append the string
        key = key.replaceAll(" ", "");
        //Removing the spaces
        HashMap<Character, Character> letters = new HashMap<>();
        //Mapping the key into a hashmap.
```

```java
        char original = 'a';
        for (int i = 0; i < key.length(); i++) {
            if (!letters.containsKey(key.charAt(i))) {
                letters.put(key.charAt(i), original++);
            }
        }
        //After the first pass all the letters of the key will be mapped with their respective original letters.
        for (int i = 0; i < message.length(); i++) {
            if (letters.containsKey(message.charAt(i))) {
                //Now replacing the letters of the message with appropriate letter according to the key
                ans.append(letters.get(message.charAt(i)));
            } else {
                ans.append(message.charAt(i));
                //This is for characters other than the letters in the key example a space " "
                //They will not be replaced by any letters hence original letter is appended into the StringBuilder
            }
        }
        return ans.toString();
    }
}
```

// https://leetcode.com/problems/decode-the-message/

Another Solution: 01

```java
class Solution {
    public String decodeMessage(String key, String message) {
        char[] map = new char[26];
        int i = 0;  // iterator.
        for (char current_ch : key.toCharArray()) {
            if (i < 26 && current_ch != ' ' && map[current_ch - 'a'] == 0) {
                map[current_ch - 'a'] = (char) (i + 'a'); // i+'a' 'b' and convering to character. 97+1 = 98(b);
                i++; // increament iterator. keep iterating.
            }
        }
        String decode = "";
        for (char current_ch : message.toCharArray()) {
            decode += current_ch == ' ' ? ' ' : map[current_ch - 'a'];
        }
        return decode;
    }
}
```

Another Solution: 02

```java
class Solution {
    public String decodeMessage(String key, String message) {
        char[] m = new char[128];
        m[' '] = ' ';
        char cur = 'a';
        for (var k : key.toCharArray())
            m[k] = m[k] != 0 ? m[k] : cur++;
```

```
        return message.chars().mapToObj(i -> String.valueOf(m[i])).collect(Collectors.joining());
    }
}
```

Another Solution: 03

```java
class Solution {
    public String decodeMessage(String key, String message) {
        Map<Character, Character> map = new HashMap<>();
        int a = 97;
        for (char c : key.toCharArray()) {
            if (c == ' ') {
                continue;
            } else {
                if (!map.containsKey(c)) {
                    map.put(c, (char) a);
                    a++;
                }
            }
        }
        StringBuilder sb = new StringBuilder();
        for (char c1 : message.toCharArray()) {
            if (c1 != ' ') {
                sb.append(map.get(c1));
            } else {
                sb.append(" ");
            }
        }
        return sb.toString();
    }
}
```

28. Problem: 1720. Decode XORed Array

There is a hidden integer array arr that consists of n non-negative integers.
It was encoded into another integer array encoded of length n - 1, such that encoded[i] = arr[i] XOR arr[i + 1]. For example, if arr = [1,0,2,1], then encoded = [1,2,3].
You are given the encoded array. You are also given an integer first, that is the first element of arr, i.e. arr[0].
Return *the original array* arr. It can be proved that the answer exists and is unique.

Example 1:
```
Input: encoded = [1,2,3], first = 1
Output: [1,0,2,1]
Explanation: If arr = [1,0,2,1], then first = 1 and encoded = [1 XOR 0, 0 XOR 2, 2 XOR 1] =
[1,2,3]
```
Example 2:
```
Input: encoded = [6,2,7,3], first = 4
Output: [4,2,0,7,4]
```

Constraints:
```
2 <= n <= 10⁴
encoded.length == n - 1
```

```
0 <= encoded[i] <= 10⁵
0 <= first <= 10⁵
```

Solution:

My Solution:

```java
class Solution_1720 {
    public int[] decode(int[] encoded, int first) {
        int first1 = first;
        int[] result = new int[encoded.length + 1];
        result[0] = first;
        for (int i = 0; i < encoded.length; i++) {
            result[i + 1] = first1 ^ encoded[i];
            first1 = first1 ^ encoded[i];
        }
        return result;
    }
}
```

// https://leetcode.com/problems/decode-xored-array/

Another Solution: 01

```java
class Solution {
    public int[] decode(int[] encoded, int first) {
        int arr[] = new int[encoded.length + 1];
        arr[0] = first;
        for (int i = 0; i < arr.length - 1; i++) {
            arr[i + 1] = arr[i] ^ encoded[i];
        }
        return arr;
    }
}
/* Explaination:
 *  A(XOR)B = C
 *  C(XOR)B = A
 */
```

Another Solution: 02

```java
class Solution {
    public int[] decode(int[] encoded, int first) {
        int x;
        int arr[] = new int[encoded.length + 1];
        arr[0] = first;
        for (int i = 0; i < encoded.length; i++) {
            x = encoded[i] ^ first;
            arr[i + 1] = x;
            first = x;
```

```
        }
        return arr;
    }
}
```

Another Solution: 03

```
//first = 1 encoded[0] = 1
// bitwise XOR ("1 + 0 = 1 , 0 + 1 = 1, 0 + 0 = 0, 1 + 1 = 1")
// 0001 ^ 0001 = 0000 = 0
// Performing an operation with xor and traditional operator

// Time: O(n)
// Space: O(n)
// explanation:
// given = [6,2,7,3]
// first = 4

// decodedArr = [4], initialized with the value of first
// 4 xor 6 = 2, add 2 into decodedArr
// decodedArr = [4, 2]
// 2 xor 2 = 0, add 0 into decodedArr
// decodedArr = [4, 2, 0]
// 0 xor 7 = 7, add 7 into decodedArr
// decodedArr = [4, 2, 0, 7]
// 7 xor 3 = 4 , add 4 into decodedArr
// final decodedArr = [4, 2, 0, 7, 4]


class Solution {
    public int[] decode(int[] encoded, int first) {
        // initalize the array with the amount of space necessary
        int[] decodedArr = new int [encoded.length+1];

        // the first value would always be the value of first
        decodedArr[0] = first;

        // keep track of the xor value
        int xor = first;

        for (int i=0; i<encoded.length; i++) {
            // '^' expression is xor
            // xor value gets updated
            xor = xor ^ encoded[i];

            // System.out.println(xor);

            // add the new decoded value to the new array
            decodedArr[i+1] = xor;
        }

        return decodedArr;
```

```
    }
}
```

27. Problem: 66. Plus One

You are given a large integer represented as an integer array `digits`, where each `digits[i]` is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's.

Increment the large integer by one and return *the resulting array of digits*.

Example 1:
```
Input: digits = [1,2,3]
Output: [1,2,4]
Explanation: The array represents the integer 123.
Incrementing by one gives 123 + 1 = 124.
Thus, the result should be [1,2,4].
```
Example 2:
```
Input: digits = [4,3,2,1]
Output: [4,3,2,2]
Explanation: The array represents the integer 4321.
Incrementing by one gives 4321 + 1 = 4322.
Thus, the result should be [4,3,2,2].
```
Example 3:
```
Input: digits = [9]
Output: [1,0]
Explanation: The array represents the integer 9.
Incrementing by one gives 9 + 1 = 10.
Thus, the result should be [1,0].
```

Constraints:
```
1 <= digits.length <= 100
0 <= digits[i] <= 9
```
`digits` does not contain any leading 0's.

Solution:

My Solution:

```java
class Solution_66 {
    public int[] plusOne(int[] digits) {

        String s = "";
        for (int i = 0; i < digits.length; i++) {
            s += digits[i];
        }

        Integer total = 1 + Integer.parseInt(s);

        String ss = String.valueOf(total);

        char[] c = ss.toCharArray();
        int[] result = new int[c.length];
```

```java
        int i = 0;
        for (char d : c) {
            result[i] = Integer.parseInt(Character.toString(d));
            i++;
        }
        return result;
    }
}
```

// https://leetcode.com/problems/plus-one/

Another Solution: 01

```java
class Solution {
    public int[] plusOne(int[] digits) {
        for(int i=digits.length-1;i>=0;i--)
        {
            if(digits[i]!=9)
            {
                digits[i]++;
                break;
            }else
                digits[i]=0;
        }
        if(digits[0]==0)
        {
            int[]res=new int [digits.length+1];
            res[0]=1;
            return res;
        }
        return digits;
    }
}
```

Another Solution: 02

```java
class Solution {
    public int[] plusOne(int[] digits) {
        for (int i = digits.length - 1; i >= 0; i--) {
            if (digits[i] < 9) {
                digits[i]++;
                return digits;
// starting from extreme right--> if array[i] is less than 9 means can be added with 1
// i.e. [ 5,8 ]-->[ 5,9 ] or
// [ 9,4 ]-->[ 9,5 ] or
// [ 6,0 ]-->[ 6,1 ]
// and will directly return array
            }
            digits[i] = 0;
// if array[i] is not less than 9, means it have to be 9 only then digit is changed to 0,
// and we again revolve around loop to check for number if less than 9 or not
```

```
// i.e. [ 5,9 ]-->[ 5,0 ]-loop->[ 6,0 ] or
// [ 1,9,9 ]-->[ 1,9,0 ]-loop->[ 1,0,0 ]-loop->[ 2,0,0 ]
// and will directly return array
        }
// if all number inside array are 9
// i.e. [ 9,9,9,9 ] than according to above loop it will become [ 0,0,0,0 ]
// but we have to make it like this [ 9,9,9,9 ]-->[ 1,0,0,0,0 ]


// to make like above we need to make new array of length--> n+1
// by default new array values are set to -->0 only
// thus just changed first value of array to 1 and return the array

        digits = new int[digits.length + 1];
        digits[0] = 1;
        return digits;
    }
}
```

Another Solution: 03

```
class Solution {
    public int[] plusOne(int[] digits) {
        for (int i = digits.length - 1; i >= 0; i--) {
            if (digits[i] != 9) {
                digits[i]++;
                break;
            } else {
                digits[i] = 0;
            }
        }
        if (digits[0] == 0) {
            int[] res = new int[digits.length + 1];
            res[0] = 1;
            return res;
        }
        return digits;
    }
}
```

26. Problem: 136. Single Number

Given a non-empty array of integers `nums`, every element appears *twice* except for one. Find that single one.
You must implement a solution with a linear runtime complexity and use only constant extra space.

Example 1:
Input: nums = [2,2,1]
Output: 1
Example 2:
Input: nums = [4,1,2,1,2]

```
Output: 4
```
Example 3:
```
Input: nums = [1]
Output: 1
```

Constraints:
```
1 <= nums.length <= 3 * 10⁴
-3 * 10⁴ <= nums[i] <= 3 * 10⁴
```
Each element in the array appears twice except for one element which appears only once.

Solution:

My Solution:

```java
class Solution_136 {
    public int singleNumber(int[] nums) {
        int result = 0;
        for (int i : nums) {
            result ^= i;
        }
        return result;
    }
}
```

// https://leetcode.com/problems/single-number/

Another Solution: 01
```java
public class Solution {
    public int singleNumber(int[] nums) {
        int result = 0;
        for (int i : nums) {
            result ^= i;
        }
        return result;
    }
}
```

Another Solution: 02
```java
//First Approach by using hashMap :-
class Solution {
    public int singleNumber(int[] nums) {
        HashMap<Integer, Integer> hash = new <Integer, Integer>HashMap();

        for (int x : nums) {
            if (hash.containsKey(x)) {       //if hashmap contains 'element', then increase correspoding val
                int count = hash.get(x);
                hash.put(x, ++count);
            } else
                hash.put(x, 1);          //if hashmap does not contain 'element', then put it in hashmap
        }
```

```java
        for (int x : hash.keySet()) {        //Iterate through 'keys' of HashMap
            if (hash.get(x) == 1)            //if 'value' for 'key' is '1', then return that 'key' element
                return x;
        }
        return -1;
    }
}
```

```java
/**
 * Time Complexity : O(nlogn) , as there is a for loop and inside for loop 'hash.containsKey'
 * Space Complexity : O(n) , as HashMap of size nearly 'n/2' is used.
 * Second Approach by using 'Sort()' method :-
 */

class Solution {
    public int singleNumber(int[] nums) {
        Arrays.sort(nums);                        //Sort the array
        for (int i = 0; i < nums.length - 1; i += 2) {
            if (nums[i] != nums[i + 1])            //Check pairwise values in array
                return nums[i];                    //if any pairwise values are different, then it is answer
        }
        return nums[nums.length - 1];     //if answer is largest value, then it will not be detected in above loop
hence return it here
    }
}
```

Day 05 -771, 1678, 2057, 2367, 1528.

25. Problem: 1528. Shuffle String

You are given a string s and an integer array indices of the same length. The string s will be shuffled such that the character at the ith position moves to indices[i] in the shuffled string.
Return *the shuffled string*.

Example 1:

```
4 5 6 7 0 2 1 3
c o d e l e e t
```

```
l e e t c o d e
0 1 2 3 4 5 6 7
```
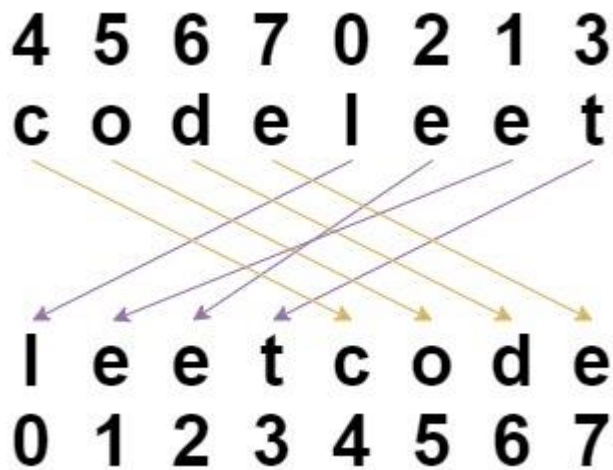
Input: s = "codeleet", indices = [4,5,6,7,0,2,1,3]
Output: "leetcode"
Explanation: As shown, "codeleet" becomes "leetcode" after shuffling.
Example 2:
Input: s = "abc", indices = [0,1,2]
Output: "abc"
Explanation: After shuffling, each character remains in its position.

Constraints:
s.length == indices.length == n
1 <= n <= 100
s consists of only lowercase English letters.
0 <= indices[i] < n
All values of indices are unique.

Solution:

My Solution:

```java
class Solution_1528 {
    public String restoreString(String s, int[] in) {
        char[] c = new char[in.length];
        for (int i = 0; i < in.length; i++)
            c[in[i]] = s.charAt(i);
        return new String(c);
    }
}
// O(n) - Time and Space

// https://leetcode.com/problems/shuffle-string/
```

Another Solution: 01

```
/**
 * Cyclic Sort -O(n) Time , O(1) Space
 * The approach is to position the index found from the indices array at the correct location
 * loop through the elements of the indices array
```

```
     * continuously swap the characters at index and i and the index of the indices array
     */

class Solution {
    public String restoreString(String s, int[] indices) {
        char[] chArray = s.toCharArray();
        int i = 0;
        while (i < indices.length) {
            if (indices[i] != i) {
                int index = indices[i];
                // swap characters
                char temp1 = chArray[index];
                chArray[index] = chArray[i];
                chArray[i] = temp1;
                // swap indexes
                int temp2 = indices[index];
                indices[index] = indices[i];
                indices[i] = temp2;
            } else {
                i++;
            }
        }
        return new String(chArray);
    }
}
```

Another Solution: 02

```
class Solution {
    public String restoreString(String s, int[] indices) {
        StringBuilder res = new StringBuilder(s);
        int l = s.length();
        for (int i = 0; i < l; i++) {
            res.setCharAt(indices[i], s.charAt(i));
        }
        return res.toString();
    }
}
```

Another Solution: 03

```
class Solution {
    public String restoreString(String s, int[] indices) {
        String answer = "";
        int a = 0;
        for (int i = 0; i < indices.length; i++) {
            if (a < indices.length) {
                if (indices[i] == a) {
                    answer += s.charAt(i);
                    a++;
                    i = -1;
                }
```

```
        }
    }
    return answer;
  }
}
```

24. Problem: 2367. Number of Arithmetic triplets

You are given a 0-indexed, strictly increasing integer array `nums` and a positive integer `diff`. A triplet `(i, j, k)` is an arithmetic triplet if the following conditions are met:
`i < j < k`,
`nums[j] - nums[i] == diff`, and
`nums[k] - nums[j] == diff`.
Return *the number of unique arithmetic triplets.*

Example 1:
```
Input: nums = [0,1,4,6,7,10], diff = 3
Output: 2
Explanation:
(1, 2, 4) is an arithmetic triplet because both 7 - 4 == 3 and 4 - 1 == 3.
(2, 4, 5) is an arithmetic triplet because both 10 - 7 == 3 and 7 - 4 == 3.
```
Example 2:
```
Input: nums = [4,5,6,7,8,9], diff = 2
Output: 2
Explanation:
(0, 2, 4) is an arithmetic triplet because both 8 - 6 == 2 and 6 - 4 == 2.
(1, 3, 5) is an arithmetic triplet because both 9 - 7 == 2 and 7 - 5 == 2.
```

Constraints:
```
3 <= nums.length <= 200
0 <= nums[i] <= 200
1 <= diff <= 50
```
`nums` is strictly increasing.

Solution:

My Solution:

```
class Solution_2367 {
  public int arithmeticTriplets(int[] nums, int diff) {
    int t = 0;

    for (int i = 0; i < nums.length; i++) {
      for (int j = 0; j < nums.length; j++) {
        for (int k = 0; k < nums.length; k++) {
          if (i != j && j != k && i < j && j < k && nums[j] - nums[i] == diff && nums[k] - nums[j] == diff) {
            t += 1;
          }
        }
      }
    }
    return t;
```

```
      }
}
```

// https://leetcode.com/problems/number-of-arithmetic-triplets/

Another Solution: 01

```java
class Solution {
    public int arithmeticTriplets(int[] nums, int diff) {
        int cnt = 0;
        Set<Integer> seen = new HashSet<>();
        for (int num : nums) {
            if (seen.contains(num - diff) && seen.contains(num - diff * 2)) {
                ++cnt;
            }
            seen.add(num);
        }
        return cnt;
    }
}
```

Another Solution: 02

```java
class Solution {
    public int arithmeticTriplets(int[] nums, int diff) {
        int count = 0;
        Map<Integer,Integer> map = new HashMap<>();
        for(int i=0;i<nums.length;i++) map.put(nums[i],i);
        for(int k : nums) count += (map.containsKey(k+diff) && map.containsKey(k+2*diff)) ? 1 : 0;
        return count;
    }
    // Time Complexity : O(log(n))
    // Space Complexity : O(n)
}
```

Another Solution: 03

```java
class Solution {
    public int arithmeticTriplets(int[] nums, int diff) {
        int count = 0;
        int pointer1 =  0 , pointer2 = 1 , pointer3 = 2;
        while(pointer3 < nums.length){
            int compare = nums[pointer2]-nums[pointer1];
            if(compare==diff){
                compare = nums[pointer3]-nums[pointer2];
                if(compare==diff){
                    pointer1++;
                    pointer2++;
                    pointer3++;
                    count++;
                }
                else if(compare<diff) pointer3++;
```

```
            else{
                pointer1++;pointer2++;
                pointer3 = Math.max(pointer3,pointer2+1);
            }
        }
        else if(compare<diff){
            pointer2++;
            pointer3 = Math.max(pointer3,pointer2+1);
        }
        else{
            pointer1++;
            pointer2 = Math.max(pointer2,pointer1+1);
            pointer3 = Math.max(pointer3,pointer2+1);
        }
    }
    return count;
}
// Time Complexity : O(n)
// Space Complexity : O(1)
}
```

23. Problem: 2057. Smallest Index with Equal Value

Given a 0-indexed integer array nums, return *the smallest index* i *of* nums *such that* i mod 10 == nums[i], *or* -1 *if such index does not exist.*

x mod y denotes the remainder when x is divided by y.

```
Example 1:
Input: nums = [0,1,2]
Output: 0
Explanation:
i=0: 0 mod 10 = 0 == nums[0].
i=1: 1 mod 10 = 1 == nums[1].
i=2: 2 mod 10 = 2 == nums[2].
All indices have i mod 10 == nums[i], so we return the smallest index 0.
Example 2:
Input: nums = [4,3,2,1]
Output: 2
Explanation:
i=0: 0 mod 10 = 0 != nums[0].
i=1: 1 mod 10 = 1 != nums[1].
i=2: 2 mod 10 = 2 == nums[2].
i=3: 3 mod 10 = 3 != nums[3].
2 is the only index which has i mod 10 == nums[i].
Example 3:
Input: nums = [1,2,3,4,5,6,7,8,9,0]
Output: -1
Explanation: No index satisfies i mod 10 == nums[i].

Constraints:
1 <= nums.length <= 100
0 <= nums[i] <= 9
```

Solution:

```java
class Solution_2057 {
    public int smallestEqual(int[] nums) {
        for (int i = 0; i < nums.length; i++) {
            if (i % 10 == nums[i]) {
                return i;
            }
        }
        return -1;
    }
}
```

// https://leetcode.com/problems/smallest-index-with-equal-value/

Another Solution: 01

```java
class Solution {
    public int smallestEqual(int[] nums) {
        int min = 101;
        for (int i = 0; i < nums.length; i++) {
            if (i % 10 == nums[i]) {
                min = Math.min(min, i);
            }
        }
        return min == 101 ? -1 : min;
    }
}
// TC: O(n), SC: O(1)
```

Another Solution: 02

```java
class Solution {
    public int smallestEqual(int[] nums) {
        int index = 0;
        for (int i = 0; i < nums.length; i++) {
            if (index == nums[i]) {
                return i;
            }
            if (++index == 10) {
                index = 0;
            }
        }
        return -1;
    }
}
```

```java
class Solution {
    public int smallestEqual(int[] nums) {
        int n = nums.length; // Length of the array
        int min = Integer.MAX_VALUE; // Assign max value to min variable
        int ans = 0;// Declare and initialize the ans variable
        for (int i = 0; i < n; i++) { // Loop for iterating the mod operator throughout the elements
            if (i % 10 == nums[i]) {
                ans = i; // Index value is assigned to variable ans
                min = Math.min(ans, min); // Find min out of the two variables
            }
        }
        if (min == Integer.MAX_VALUE) return -1; // If min value remains unchanged
        return min; // Else return the min index
    }
}
```

22. Problem: 1678. Goal Parser Interpretation

You own a Goal Parser that can interpret a string `command`. The `command` consists of an alphabet of `"G"`, `"()"` and/or `"(al)"` in some order. The Goal Parser will interpret `"G"` as the string `"G"`, `"()"` as the string `"o"`, and `"(al)"` as the string `"al"`. The interpreted strings are then concatenated in the original order.
Given the string `command`, return *the Goal Parser's interpretation of* `command`.

Example 1:
```
Input: command = "G()(al)"
Output: "Goal"
Explanation: The Goal Parser interprets the command as follows:
G -> G
() -> o
(al) -> al
The final concatenated result is "Goal".
```
Example 2:
```
Input: command = "G()()()()(al)"
Output: "Gooooal"
```
Example 3:
```
Input: command = "(al)G(al)()()G"
Output: "alGalooG"
```

Constraints:
```
1 <= command.length <= 100
```
`command` consists of `"G"`, `"()"`, and/or `"(al)"` in some order.

Solution:

My Solution:

```java
class Solution_1678 {
    public String interpret(String input) {
```

```
        input = input.replaceAll("\\(\\)","o");
        input = input.replaceAll("\\(al\\)","al");
        return input;
    }
}
```

// https://leetcode.com/problems/goal-parser-interpretation/

```java
class Solution {
    public String interpret(String command) {
        char[] str = command.toCharArray();
        String fi="";
        for(int i=0;i<str.length;i++){
            if(str[i]=='G'){
                fi=fi+str[i];
            }
            else if(str[i]=='(' && str[i+1]==')'){
                fi=fi+'o';
            }
            else if(str[i]=='(' && str[i+1]=='a'){
                fi= fi+"al";
            }
        }
        return fi;
    }
}
```

```java
class Solution {
    public String interpret(String command) {
        StringBuilder S = new StringBuilder(command.length());
        for(int i = 0 ; i < command.length() ; i++){
            if(command.charAt(i)=='G') S.append('G');
            if(command.charAt(i)=='('){
                if(command.charAt(i+1)==')') {S.append('o'); i++;}
                else{S.append("al"); i = i + 3;}
            }
        }
        return S.toString();
    }
}
```

```java
class Solution {
    public String interpret(String command) {
        return command.replace("()", "o").replace("(al)", "al");
```

```
        }
}
```

21. Problem: 771. Jewels and Stone

You're given strings `jewels` representing the types of stones that are jewels, and `stones` representing the stones you have.
Each character in `stones` is a type of stone you have. You want to know how many of the stones you have are also jewels.
Letters are case sensitive, so `"a"` is considered a different type of stone from `"A"`.

Example 1:
```
Input: jewels = "aA", stones = "aAAbbbb"
Output: 3
```
Example 2:
```
Input: jewels = "z", stones = "ZZ"
Output: 0
```

Constraints:
```
1 <= jewels.length, stones.length <= 50
```
`jewels` and `stones` consist of only English letters.
All the characters of `jewels` are unique.

Solution:

My Solution:

```java
class Solution_771 {
    public int numJewelsInStones(String jewels, String stones) {
        int s = 0;
        for(int i = 0 ; i < jewels.length() ; i++){
            for(int j = 0 ; j < stones.length() ; j++){
                if (jewels.charAt(i) == stones.charAt(j)){
                    s += 1;
                    //System.out.println(jewels.charAt(i) + " " + stones.charAt(j));
                }
            }
        }
        return s;
    }
}
```

// https://leetcode.com/problems/jewels-and-stones/submissions/

Another Solution: 01

```java
class Solution {
    public int numJewelsInStones(String jewels, String stones) {
        int num = 0;
        for (int i = 0 ; i < stones.length(); i ++) {
            if(jewels.indexOf(stones.charAt(i)) != -1) {
                num++;
            }
        }
        return num;
```

```
    }
}
```

```
class Solution {
    public int numJewelsInStones(String J, String S) {
        if(J.isEmpty()) return 0;
        int cnt = 0;
        for(int i = 0 ; i < S.length() ; i++) {
            if(J.indexOf(S.charAt(i)) != -1) cnt++;
        }
        return cnt;
    }
}
```

```
class Solution {
    public int numJewelsInStones(String jewels, String stones) {
        int count=0;
        for(char c:stones.toCharArray()){
            if(jewels.indexOf(c)!=-1)count++;
        }
        return count;
    }
}
```

Day 04 -1470, 1672, 1431, 2367, 2057.

20. Problem: 2057. Smallest Index with equal value

Given a **0-indexed** integer array nums, return *the **smallest index** i of nums such that i mod 10 == nums[i], or -1 if such index does not exist.*
x mod y denotes the **remainder** when x is divided by y.

Example 1:
Input: nums = [0,1,2]
Output: 0
Explanation:
i=0: 0 mod 10 = 0 == nums[0].
i=1: 1 mod 10 = 1 == nums[1].
i=2: 2 mod 10 = 2 == nums[2].
All indices have i mod 10 == nums[i], so we return the smallest index 0.
Example 2:
Input: nums = [4,3,2,1]
Output: 2
Explanation:

```
i=0: 0 mod 10 = 0 != nums[0].
i=1: 1 mod 10 = 1 != nums[1].
i=2: 2 mod 10 = 2 == nums[2].
i=3: 3 mod 10 = 3 != nums[3].
2 is the only index which has i mod 10 == nums[i].
```
Example 3:
Input: nums = [1,2,3,4,5,6,7,8,9,0]
Output: -1
Explanation: No index satisfies i mod 10 == nums[i].

Constraints:
```
1 <= nums.length <= 100
0 <= nums[i] <= 9
```

Solution:

My Solution:

```java
package O1_easy.day_04;

public class O20_SmallestIndexWithEqualValue_2057 {
    public static void main(String[] args) {
        Solution_2057 solution_2057 = new Solution_2057();
        int[] arr = {0, 1, 2};
        int[] arr2 = {4, 3, 2, 1};
        int[] arr3 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};
        System.out.println(solution_2057.smallestEqual(arr));
        System.out.println(solution_2057.smallestEqual(arr2));
        System.out.println(solution_2057.smallestEqual(arr3));
    }
}

class Solution_2057 {
    public int smallestEqual(int[] nums) {
        for (int i = 0; i < nums.length; i++) {
            if (i % 10 == nums[i]) {
                return i;
            }
        }
        return -1;
    }
}

// https://leetcode.com/problems/smallest-index-with-equal-value/
```

Another Solution: 01

```java
class Solution {
    public int smallestEqual(int[] a) {
        for (int d1 = 0; d1 <= 9; d1++)
            for (int d2 = 0; d2 <= 9 && 10 * d1 + d2 < a.length; d2++)
```

```
        if (d2 == a[d1 * 10 + d2]) return d1 * 10 + d2;
        return -1;
    }
}
```

```
class Solution {
    public int smallestEqual(int[] n) {
        int k = n.length;
        for (int i = 0; i < k; i++) {
            if (i % 10 == n[i]) return i;
        }
        return -1;
    }
}
```

```
class Solution {
    public int smallestEqual(int[] nums) {
        for (int i = 0; i < nums.length; i++) {
            if (i % 10 == nums[i]) {
                return i;
            }
        }
        return -1;
    }
}
```

19. Problem: 2367. Number of Arithmetic Triplets

You are given a **0-indexed**, **strictly increasing** integer array `nums` and a positive integer `diff`. A triplet `(i, j, k)` is an **arithmetic triplet** if the following conditions are met:
`i < j < k`,
`nums[j] - nums[i] == diff`, and
`nums[k] - nums[j] == diff`.
Return *the number of unique **arithmetic triplets***.

Example 1:
```
Input: nums = [0,1,4,6,7,10], diff = 3
Output: 2
Explanation:
(1, 2, 4) is an arithmetic triplet because both 7 - 4 == 3 and 4 - 1 == 3.
(2, 4, 5) is an arithmetic triplet because both 10 - 7 == 3 and 7 - 4 == 3.
```
Example 2:
```
Input: nums = [4,5,6,7,8,9], diff = 2
Output: 2
Explanation:
(0, 2, 4) is an arithmetic triplet because both 8 - 6 == 2 and 6 - 4 == 2.
(1, 3, 5) is an arithmetic triplet because both 9 - 7 == 2 and 7 - 5 == 2.
```

Constraints:

```
3 <= nums.length <= 200
0 <= nums[i] <= 200
1 <= diff <= 50
```

nums is **strictly** increasing.

Solution:

My Solution:

```java
import java.util.HashSet;
import java.util.Set;
import java.util.stream.IntStream;

public class O19_NumberOfArithmeticTriplets_2367 {
    public static void main(String[] args) {
        Solution_2367 solution_2367 = new Solution_2367();
        int[] num = {0, 1, 4, 6, 7, 10};
        int[] num2 = {4, 5, 6, 7, 8, 9};
        System.out.println(solution_2367.arithmeticTriplets(num, 3));
        System.out.println(solution_2367.arithmeticTriplets(num2, 2));
    }
}

class Solution_2367 {
    public int arithmeticTriplets(int[] nums, int diff) {
        int t = 0;
        for (int i = 0; i < nums.length; i++) {
            for (int j = 0; j < nums.length; j++) {
                for (int k = 0; k < nums.length; k++) {
                    if (i != j && j != k && i < j && j < k && nums[j] - nums[i] == diff && nums[k] - nums[j] == diff) {
                        t += 1;
                    }
                }
            }
        }
        return t;
    }
}
```

// https://leetcode.com/problems/number-of-arithmetic-triplets/

Another Solution: 01

```java
class Solution {
    public int arithmeticTriplets(int[] nums, int diff) {
        int cnt = 0;
        Set<Integer> seen = new HashSet<>();
        for (int num : nums) {
            if (seen.contains(num - diff) && seen.contains(num - diff * 2)) {
```

```
            ++cnt;
        }
        seen.add(num);
    }
    return cnt;
    }
}
```

```
class Solution {
    public int arithmeticTriplets(int[] nums, int diff) {
        int count = 0;
        for (int i = 0; i < nums.length - 2; i++) {
            for (int j = i + 1; j < nums.length - 1; j++) {
                for (int k = j + 1; k < nums.length; k++) {
                    if (nums[j] - nums[i] == diff && nums[k] - nums[j] == diff) count++;
                }
            }
        }
        return count;
    }
}
```

```
class Solution {
    public int arithmeticTriplets(int[] nums, int diff) {
        Set<Integer> set = new HashSet<>();
        for (int num : nums) {
            set.add(num);
        }
        int ans = 0;
        for (int num : nums) {
            if ((set.contains(num + diff) && set.contains(num + 2 * diff))) {
                ans++;
            }
        }
        return ans;
    }
}
```

18. Problem: 1431. Kids with Greatest Number Of Candies

There are n kids with candies. You are given an integer array `candies`, where each `candies[i]` represents the number of candies the ith kid has, and an integer `extraCandies`, denoting the number of extra candies that you have.

Return *a boolean array* `result` *of length* n*, where* `result[i]` *is* `true` *if, after giving the* i^{th} *kid all the* `extraCandies`*, they will have the **greatest** number of candies among all the kids, or* `false` *otherwise.*

Note that **multiple** kids can have the **greatest** number of candies.

Example 1:
Input: candies = [2,3,5,1,3], extraCandies = 3
Output: [true,true,true,false,true]
Explanation: If you give all extraCandies to:
- Kid 1, they will have 2 + 3 = 5 candies, which is the greatest among the kids.
- Kid 2, they will have 3 + 3 = 6 candies, which is the greatest among the kids.
- Kid 3, they will have 5 + 3 = 8 candies, which is the greatest among the kids.
- Kid 4, they will have 1 + 3 = 4 candies, which is not the greatest among the kids.
- Kid 5, they will have 3 + 3 = 6 candies, which is the greatest among the kids.

Example 2:
Input: candies = [4,2,1,1,2], extraCandies = 1
Output: [true,false,false,false,false]
Explanation: There is only 1 extra candy.
Kid 1 will always have the greatest number of candies, even if a different kid is given the extra candy.

Example 3:
Input: candies = [12,1,12], extraCandies = 10
Output: [true,false,true]

Constraints:
```
n == candies.length
2 <= n <= 100
1 <= candies[i] <= 100
1 <= extraCandies <= 50
```

Solution:

My Solution:

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

public class O18_KidsWithGreatestNumberOfCandies_1431 {
    public static void main(String[] args) {
        Solution_1431 solution_1431 = new Solution_1431();
        int[] arr = {2, 3, 5, 1, 3};
        int[] arr2 = {4, 2, 1, 1, 2};
        int[] arr3 = {12, 1, 12};
        System.out.println(solution_1431.kidsWithCandies(arr, 3));
        System.out.println(solution_1431.kidsWithCandies(arr2, 1));
        System.out.println(solution_1431.kidsWithCandies(arr3, 10));
    }
}

class Solution_1431 {
    public List<Boolean> kidsWithCandies(int[] candies, int extraCandies) {
        List<Boolean> result = new ArrayList<>();
        for (int i = 0; i < candies.length; i++) {
            boolean b = false;
            for (int j = 0; j < candies.length; j++) {
                if (i != j && candies[i] + extraCandies < candies[j]) {
```

```
                b = false;
                break;
            } else {
                b = true;
            }
        }
    }
    result.add(b);
    }
    return result;
    }
}
```

// https://leetcode.com/problems/kids-with-the-greatest-number-of-candies/

Another Solution: 01

```
class Solution {
    public List<Boolean> kidsWithCandies(int[] candies, int extraCandies) {
        int max = Arrays.stream(candies).max().getAsInt();
        return Arrays.stream(candies).mapToObj(candy -> candy + extraCandies >=
max).collect(Collectors.toList());
    }
}
```

Another Solution: 02

```
class Solution {
    public List<Boolean> kidsWithCandies(int[] cs, int e) {
        int max = 0;
        List<Boolean> res = new ArrayList<>();
        for (int n : cs) max = Math.max(n, max);
        for (int n : cs) res.add(n + e >= max);
        return res;
    }
}
```

Another Solution: 03

```
class Solution {
    public List<Boolean> kidsWithCandies(int[] candies, int extraCandies) {
        List<Boolean> ans = new ArrayList<>(candies.length);
        int max = 0;
        for (int candy : candies) {
            max = Math.max(candy, max);
        }
        for (int candy : candies) {
            ans.add(candy + extraCandies >= max);
        }
        return ans;
    }
```

}

17. Problem: 1672. Richest Customer Wealth

You are given an `m x n` integer grid `accounts` where `accounts[i][j]` is the amount of money the `i`th customer has in the `j`th bank. Return *the **wealth** that the richest customer has.*

A customer's **wealth** is the amount of money they have in all their bank accounts. The richest customer is the customer that has the maximum **wealth**.

Example 1:
```
Input: accounts = [[1,2,3],[3,2,1]]
Output: 6
Explanation:
1st customer has wealth = 1 + 2 + 3 = 6
2nd customer has wealth = 3 + 2 + 1 = 6
Both customers are considered the richest with a wealth of 6 each, so return 6.
```
Example 2:
```
Input: accounts = [[1,5],[7,3],[3,5]]
Output: 10
Explanation:
1st customer has wealth = 6
2nd customer has wealth = 10
3rd customer has wealth = 8
The 2nd customer is the richest with a wealth of 10.
```
Example 3:
```
Input: accounts = [[2,8,7],[7,1,3],[1,9,5]]
Output: 17
```

Constraints:
```
m == accounts.length
n == accounts[i].length
1 <= m, n <= 50
1 <= accounts[i][j] <= 100
```

Solution:

My Solution:

```java
import java.util.Arrays;

public class O17_RichestCustomerWealth_1672 {
    public static void main(String[] args) {
        Solution_1672 solution_1672 = new Solution_1672();
        int[][] arr = {{1, 2, 3}, {3, 2, 1}};
        int[][] arr2 = {{1, 5}, {7, 3}, {3, 5}};
        int[][] arr3 = {{2, 8, 7}, {7, 1, 3}, {1, 9, 5}};
        System.out.println(solution_1672.maximumWealth(arr));
        System.out.println(solution_1672.maximumWealth(arr2));
        System.out.println(solution_1672.maximumWealth(arr3));
    }
}
```

```java
class Solution_1672 {
    public int maximumWealth(int[][] accounts) {
        int max = 0;
        for (int i = 0; i < accounts.length; i++) {
            int a = wealth(accounts[i]);
            if (a > max) {
                max = a;
            }
        }
        return max;
    }

    public int wealth(int[] a) {
        int sum = 0;
        for (int i = 0; i < a.length; i++) {
            sum += a[i];
        }
        return sum;
    }
}
```

// https://leetcode.com/problems/richest-customer-wealth/submissions/

Another Solution: 01

```java
class Solution {
    public int maximumWealth(int[][] accounts) {
        int temp = 0;
        for (int i = 0; i < accounts.length; i++) {
            int sum = 0;
            for (int j = 0; j < accounts[i].length; j++) {
                sum = sum + accounts[i][j];
            }
            if (sum > temp) {
                temp = sum;
            }
        }
        return temp;
    }
}
```

Another Solution: 02

```java
class Solution {
    public int maximumWealth(int[][] accounts) {
        return Arrays.stream(accounts)
                .mapToInt(i -> Arrays.stream(i).sum())
                .max()
                .getAsInt();
    }
}
```

```java
class Solution {
    public int maximumWealth(int[][] accounts) {
        var maxWealth = 0;
        for (var customer : accounts) {
            maxWealth = Math.max(maxWealth, Arrays.stream(customer).sum());
        }
        return maxWealth;
    }
}
```

16. Problem: 1470. Shuffle the array

Given the array `nums` consisting of 2n elements in the form $[x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n]$.
Return the array in the form $[x_1, y_1, x_2, y_2, \ldots, x_n, y_n]$.

Example 1:
```
Input: nums = [2,5,1,3,4,7], n = 3
Output: [2,3,5,4,1,7]
Explanation: Since x₁=2, x₂=5, x₃=1, y₁=3, y₂=4, y₃=7 then the answer is [2,3,5,4,1,7].
```
Example 2:
```
Input: nums = [1,2,3,4,4,3,2,1], n = 4
Output: [1,4,2,3,3,2,4,1]
```
Example 3:
```
Input: nums = [1,1,2,2], n = 2
Output: [1,2,1,2]
```

Constraints:
```
1 <= n <= 500
nums.length == 2n
1 <= nums[i] <= 10^3
```

Solution:

My Solution:

```java
import java.util.Arrays;
import java.util.stream.IntStream;

public class O16_ShuffleTheString_1470 {
    public static void main(String[] args) {
        Solution_1470 solution_1470 = new Solution_1470();
        int[] arr = {2, 5, 1, 3, 4, 7};
        int[] arr2 = {1, 2, 3, 4, 4, 3, 2, 1};
        int[] arr3 = {1, 1, 2, 2};
        System.out.println(Arrays.toString(solution_1470.shuffle(arr, 3)));
        System.out.println(Arrays.toString(solution_1470.shuffle(arr2, 4)));
        System.out.println(Arrays.toString(solution_1470.shuffle(arr3, 2)));
    }
}
```

```java
class Solution_1470 {
    public int[] shuffle(int[] nums, int n) {
        int[] arrR = new int[nums.length];
        int t = 0;
        for (int k = 0; k < nums.length; k += 2) {
            arrR[k] = nums[t];
            t++;
        }
        int t2 = n;
        for (int l = 1; l < nums.length; l += 2) {
            arrR[l] = nums[t2];
            t2++;
        }
        return arrR;
    }
}
```

// https://leetcode.com/problems/shuffle-the-array/submissions/

Another Solution: 01

```java
class Solution {
    public int[] shuffle(int[] nums, int n) {
        int len = nums.length;
        for (int i = n; i < len; i++) {
            nums[i] = (nums[i] << 10) | nums[i - n];
        }
        int index = 0;
        for (int i = n; i < len; i++, index += 2) {
            nums[index] = nums[i] & 1023;
            nums[index + 1] = nums[i] >>> 10;
        }
        return nums;
    }
}
```

Another Solution: 02

```java
class Solution {
    public int[] shuffle(int[] nums, int n) {
        return IntStream.range(0, 2 * n).map(i -> nums[i / 2 + i % 2 * n]).toArray();
    }
}
```

Another Solution: 03

```java
class Solution {
    public int[] shuffle(int[] nums, int n) {
        int[] res = new int[2 * n];
        for (int i = 0, j = n, idx = 0; idx < res.length; i++, j++) {
            res[idx++] = nums[i];
```

```
            res[idx++] = nums[j];
        }
        return res;
    }
}
```

Day 03 -2114, 1, 24, 415, 709.

15. Problem: 709. To Lower Case

Given a string `s`, return the string after replacing every uppercase letter with the same lowercase letter.

Example 1:
```
Input: s = "Hello"
Output: "hello"
```
Example 2:
```
Input: s = "here"
Output: "here"
```
Example 3:
```
Input: s = "LOVELY"
Output: "lovely"
```

Constraints:
```
1 <= s.length <= 100
```
`s` consists of printable ASCII characters.

Solution:

My Solution:

```java
public class O15_ToLowerCase_709 {
    public static void main(String[] args) {
        Solution_709 solution_709 = new Solution_709();
        String s = "Hello";
        System.out.println(solution_709.interpret(s));
    }
}

class Solution_709 {
    public String interpret(String command) {
        return command.toLowerCase();
    }
}

// https://leetcode.com/problems/to-lower-case/
```

Another Solution: 01

```java
class Solution_709 {
    public String toLowerCase(String str) {
        char[] a = str.toCharArray();
        for (int i = 0; i < a.length; i++)
            if ('A' <= a[i] && a[i] <= 'Z')
                a[i] = (char) (a[i] - 'A' + 'a');
        return new String(a);
    }
}
```

```java
class Solution_709 {
    public String toLowerCase(String s) {
        char[] a = s.toCharArray();
IntStream.range(0, a.length).filter(i -> 'A' <= a[i] && a[i] <= 'Z').forEach(i -> a[i] = (char) (a[i] - 'A' + 'a'));
        return new String(a);
    }
}
```

```java
class Solution {
    public String toLowerCase(String s) {
        StringBuilder str = new StringBuilder();
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) >= 65 && s.charAt(i) <= 90) {
                str.append((char) (s.charAt(i) + 32));
            } else {
                str.append(s.charAt(i));
            }
        }
        return str.toString();
    }
}
```

14. Problem: 415. Add String

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 <= 10⁴
```

num1 and num2 consist of only digits.

num1 and num2 don't have any leading zeros except for the zero itself.

Solution:

My Solution:

```java
import java.math.BigInteger;

public class O14_AddString_415 {
    public static void main(String[] args) {
        Solution_415 solution_415 = new Solution_415();
        String s = "654615";
        String s2 = "654615";
        System.out.println(solution_415.addStrings(s, s2));
    }
}

class Solution_415 {
    public String addStrings(String num1, String num2) {
        BigInteger bigInteger = new BigInteger(num1);
        BigInteger bigInteger2 = new BigInteger(num2);
        return String.valueOf(bigInteger2.add(bigInteger));
    }
}
```

// https://leetcode.com/problems/add-strings/submissions/

Another Solution: 01

```java
class Solution {
    public String addStrings(String num1, String num2) {
        int i = num1.length() - 1, j = num2.length() - 1, carry = 0;
        StringBuilder sb = new StringBuilder();

        while (i >= 0 || j >= 0 || carry != 0) {
            if (i >= 0) carry += num1.charAt(i--) - '0';
            if (j >= 0) carry += num2.charAt(j--) - '0';
            sb.append(carry % 10);
            carry /= 10;
        }
        return sb.reverse().toString();
    }
}
```

Another Solution: 02

```java
class Solution {
    public String addStrings(String num1, String num2) {
        int carry = 0;
        int i = num1.length() - 1, j = num2.length() - 1;
        StringBuilder sb = new StringBuilder();
        while (i >= 0 || j >= 0) {
            int n1 = 0, n2 = 0;
            if (i >= 0) {
                n1 = num1.charAt(i) - '0';
            }
            if (j >= 0) {
                n2 = num2.charAt(j) - '0';
            }
            int sum = n1 + n2 + carry;
            carry = sum / 10;
            sb.append(sum % 10);
            i--;
            j--;
        }
        if (carry != 0) {
            sb.append(carry);
        }
        return sb.reverse().toString();
    }
}
```

Another Solution: 03

```java
public class Solution {
    public String addStrings(String num1, String num2) {
        int i = num1.length() - 1;
        int j = num2.length() - 1;
        int carry = 0;
        char[] num1Array = num1.toCharArray();
        char[] num2Array = num2.toCharArray();
        StringBuilder sb = new StringBuilder();
        while (i >= 0 || j >= 0 || carry == 1) {
            int a = i >= 0 ? (num1Array[i--] - '0') : 0;
            int b = j >= 0 ? (num2Array[j--] - '0') : 0;
            int sum = a + b + carry;
            sb.insert(0, sum % 10);
            carry = sum / 10;
        }
        return sb.toString();
    }
}
```

```
/**
 * Complexity Analysis
 * Time Complexity: `O(m + n)` (Average Case) and `O(m + n)` (Worst Case) where `m` and `n` are the total
```

13. Problem: 26. Remove Duplicates From sorted array

Given an integer array `nums` sorted in **non-decreasing order**, remove the duplicates **in-place** such that each unique element appears only **once**. The **relative order** of the elements should be kept the **same**.

Since it is impossible to change the length of the array in some languages, you must instead have the result be placed in the **first part** of the array `nums`. More formally, if there are `k` elements after removing the duplicates, then the first `k` elements of `nums` should hold the final result. It does not matter what you leave beyond the first `k` elements.

Return `k` *after placing the final result in the first* `k` *slots of* `nums`.

Do **not** allocate extra space for another array. You must do this by **modifying the input array in-place** with O(1) extra memory.

Custom Judge:
The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int[] expectedNums = [...]; // The expected answer with correct length

int k = removeDuplicates(nums); // Calls your implementation

assert k == expectedNums.length;
for (int i = 0; i < k; i++) {
    assert nums[i] == expectedNums[i];
}
```

If all assertions pass, then your solution will be **accepted**.

Example 1:
Input: nums = [1,1,2]
Output: 2, nums = [1,2,_]
Explanation: Your function should return k = 2, with the first two elements of nums being 1 and 2 respectively.
It does not matter what you leave beyond the returned k (hence they are underscores).

Example 2:
Input: nums = [0,0,1,1,1,2,2,3,3,4]
Output: 5, nums = [0,1,2,3,4,_,_,_,_,_]
Explanation: Your function should return k = 5, with the first five elements of nums being 0, 1, 2, 3, and 4 respectively.
It does not matter what you leave beyond the returned k (hence they are underscores).

Constraints:
1 <= nums.length <= 3 * 10⁴
-100 <= nums[i] <= 100

nums is sorted in **non-decreasing** order.

Solution:

My Solution:

```java
import java.util.*;

public class O13_RemoveDuplicateFromsortedArray_26 {
    public static void main(String[] args) {
        Solution solution = new Solution();
        int[] arr = {1, 1, 2};
        int[] arr2 = {0, 0, 1, 1, 1, 2, 2, 3, 3, 4};
        System.out.println(solution.removeDuplicates(arr));
        System.out.println(solution.removeDuplicates(arr2));
    }
}

class Solution {
    public int removeDuplicates(int[] nums) {
        int[] temp = new int[nums.length];
        int j = 0;
        for (int i = 0; i < nums.length - 1; i++) {
            if (nums[i] != nums[i + 1]) {
                temp[j++] = nums[i];
            }
        }

        temp[j++] = nums[nums.length - 1];

        // Changing original array
        for (int i = 0; i < j; i++) {
            nums[i] = temp[i];
        }
        return j;
    }
}
```

// https://leetcode.com/problems/remove-duplicates-from-sorted-array/

Another Solution: 01

```java
class Solution {
    public int removeDuplicates(int[] nums) {
        int i = 0;
        for (int n : nums)
            if (i == 0 || n > nums[i - 1])
                nums[i++] = n;
        return i;
    }
}
```

// And to not need the i ==0check in the loop:

```java
class Solution {
    public int removeDuplicates(int[] nums) {
        int i = nums.length > 0 ? 1 : 0;
        for (int n : nums)
            if (n > nums[i - 1])
                nums[i++] = n;
        return i;
    }
}
```

Another Solution: 02

```java
class Solution {
    public int removeDuplicates(int[] nums) {
        int pos = 0;
        for (int num : nums) {
            if (nums[pos] != num) {
                nums[++pos] = num;
            }
        }
        return pos + 1;
    }
}
```

Another Solution: 03

```java
class Solution {
    public int removeDuplicates(int[] A) {
        if (A.length == 0) return 0;
        int j = 0;
        for (int i = 0; i < A.length; i++)
            if (A[i] != A[j]) A[++j] = A[i];
        return ++j;
    }
}
```

Duplicate count + Print Unique value using Hashmap

```java
class Solution {
   public int removeDuplicates(int[] nums) {

HashMap<Integer, Integer> hashmap = new HashMap<>();

int duplicateCount = 0;
for (int i = 1; i < nums.length; i++) {
   if (nums[i] == nums[i - 1]) {
      duplicateCount++;
   }
   nums[i - duplicateCount] = nums[i];

   hashmap.put(nums[i], i); // Value of nums insert in Hashmap as key and index as value
   // hashmap.put(i, nums[i]); this will not work , key can not be duplicate, value can
}
System.out.println(hashmap.keySet()); // output:  [0, 1, 2, 3, 4]
System.out.println(hashmap.values()); // output:  [1, 4, 6, 8, 9]
return nums.length - duplicateCount;

}
```

12. Problem: 01. Two Sum

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to `target`*.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice. You can return the answer in any order.

Example 1:
```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
```
Example 2:
```
Input: nums = [3,2,4], target = 6
Output: [1,2]
```
Example 3:
```
Input: nums = [3,3], target = 6
Output: [0,1]
```

Constraints:
```
2 <= nums.length <= 10⁴
-10⁹ <= nums[i] <= 10⁹
-10⁹ <= target <= 10⁹
```
Only one valid answer exists.

Follow-up: Can you come up with an algorithm that is less than O(n2) time complexity?

Solution:

My Solution:

```java
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

public class O12_TwoSum_1 {
    public static void main(String[] args) {
        Solution_1 solution_1 = new Solution_1();
        int[] arr = {2, 7, 11, 15};
        int[] arr2 = {3, 2, 4};
        System.out.println(Arrays.toString(solution_1.twoSum(arr, 9)));
        System.out.println(Arrays.toString(solution_1.twoSum(arr2, 6)));
    }
}


class Solution_1 {
    public int[] twoSum(int[] nums, int target) {
        int[] result = new int[2];
        for (int i = 0; i < nums.length; i++) {
            for (int j = 0; j < nums.length; j++) {
                if (i != j && nums[i] + nums[j] == target) {
                    result[0] = i;
                    result[1] = j;
                    break;
                }
            }
        }
        return result;
    }
}
```

// https://leetcode.com/problems/two-sum/submissions/

Another Solution: 01

```java
class Solution {
    public int[] twoSum(int[] numbers, int target) {
        int[] result = new int[2];
        Map<Integer, Integer> map = new HashMap<Integer, Integer>();
        for (int i = 0; i < numbers.length; i++) {
            if (map.containsKey(target - numbers[i])) {
                result[1] = i;
                result[0] = map.get(target - numbers[i]);
                return result;
            }
            map.put(numbers[i], i);
        }
        return result;
```

```
    }
}
```

```java
class Solution {
    public int[] twoSum(int[] nums, int target) {
        HashMap<Integer, Integer> seen = new HashMap<>();
        for (int i = 0; i < nums.length; ++i) {
            int b = nums[i], a = target - b;
            if (seen.containsKey(a)) return new int[]{seen.get(a), i}; // Found pair of (a, b), so that a + b = target
            seen.put(b, i);
        }
        return new int[]{};
    }
}
```

```java
class Solution {
    //O(nlogn)
    public int[] twoSum_n2(int[] nums, int target) {
        if (nums == null) return null;
        int[] nums2 = Arrays.copyOf(nums, nums.length);
        Arrays.sort(nums2);
        int a = 0, b = 0;
        int start = 0, end = nums2.length - 1;
        //find two nums
        while (start < end) {
            int sum = nums2[start] + nums2[end];
            if (sum < target) start++;
            else if (sum > target) end--;
            else {
                a = nums2[start];
                b = nums2[end];
                break;
            }
        }
        //find the index of two numbers
        int[] res = new int[2];
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] == a) {
                res[0] = i;
                break;
            }
        }
        if (a != b) {
            for (int i = 0; i < nums.length; i++) {
                if (nums[i] == b) {
                    res[1] = i;
                    break;
                }
```

```
            }
        } else {
            for (int i = 0; i < nums.length; i++) {
                if (nums[i] == b && i != res[0]) {
                    res[1] = i;
                    break;
                }
            }
        }
        return res;
    }
}
```

11. Problem: 2114. Maximum Number of Words found in Sentence

A sentence is a list of words that are separated by a single space with no leading or trailing spaces.
You are given an array of strings `sentences`, where each `sentences[i]` represents a single sentence.
Return *the maximum number of words that appear in a single sentence*.

Example 1:
```
Input: sentences = ["alice and bob love leetcode", "i think so too", "this is great thanks very
much"]
Output: 6
Explanation:
- The first sentence, "alice and bob love leetcode", has 5 words in total.
- The second sentence, "i think so too", has 4 words in total.
- The third sentence, "this is great thanks very much", has 6 words in total.
Thus, the maximum number of words in a single sentence comes from the third sentence, which has 6
words.
```
Example 2:
```
Input: sentences = ["please wait", "continue to fight", "continue to win"]
Output: 3
Explanation: It is possible that multiple sentences contain the same number of words.
In this example, the second and third sentences (underlined) have the same number of words.
```

Constraints:
```
1 <= sentences.length <= 100
1 <= sentences[i].length <= 100
```
`sentences[i]` consists only of lowercase English letters and ' ' only.
`sentences[i]` does not have leading or trailing spaces.
All the words in `sentences[i]` are separated by a single space.

Solution:

My Solution:

```java
import java.util.Arrays;
import java.util.stream.Stream;

public class O11_MaxNumberOfWordInSentence_2114 {
    public static void main(String[] args) {
```

```java
        Solution_2114 solution_2114 = new Solution_2114();
        String[] arr = {"alice and bob love leetcode", "i think so too", "this is great thanks very much"};
        String[] arr2 = {"please wait", "continue to fight", "continue to win"};
        System.out.println((solution_2114.mostWordsFound(arr)));
        System.out.println((solution_2114.mostWordsFound(arr2)));
    }
}

class Solution_2114 {
    public int mostWordsFound(String[] sentences) {
        int max = 0;
        for (int i = 0; i < sentences.length; i++) {
            int c = most(sentences[i]);
            if (c > max) {
                max = c;
            }
        }
        return max;
    }

    public int most(String s) {
        int m = 0;
        String[] strings = s.split(" ");
        m = strings.length;
        return m;
    }
}

// https://leetcode.com/problems/maximum-number-of-words-found-in-sentences/
```

Another Solution: 01

```java
class Solution {
    public int mostWordsFound(String[] sentences) {
        return Stream.of(sentences).mapToInt(s -> s.split(" ").length).max().getAsInt();
    }
}
```

Another Solution: 02

```java
class Solution {
    public int mostWordsFound(String[] sentences) {
        int max = 0;
        for (String s : sentences) {
            max = Math.max(max, s.split(" ").length);
        }
        return max;
    }
}
```

Another Solution: 03

```java
class Solution {
    public int mostWordsFound(String[] sentences) {
        int max = 0;
        for (String s : sentences) {
            int x = s.indexOf(" "), count = 1;
            while (x != -1) {
                count++;
                x = s.indexOf(" ", x + 1);
            }
            max = Math.max(count, max);
        }
        return max;
    }
}
```

Day 02 -1929, 1920, 1512, 1365, 1816.

10. Problem: 1816. Truncate Sentence

A **sentence** is a list of words that are separated by a single space with no leading or trailing spaces. Each of the words consists of **only** uppercase and lowercase English letters (no punctuation).

For example, `"Hello World"`, `"HELLO"`, and `"hello world hello world"` are all sentences.

You are given a sentence `s` and an integer `k`. You want to **truncate** `s` such that it contains only the **first** `k` words. Return `s` *after **truncating** it.*

Example 1:
```
Input: s = "Hello how are you Contestant", k = 4
Output: "Hello how are you"
Explanation:
The words in s are ["Hello", "how" "are", "you", "Contestant"].
The first 4 words are ["Hello", "how", "are", "you"].
Hence, you should return "Hello how are you".
```
Example 2:
```
Input: s = "What is the solution to this problem", k = 4
Output: "What is the solution"
Explanation:
The words in s are ["What", "is" "the", "solution", "to", "this", "problem"].
The first 4 words are ["What", "is", "the", "solution"].
Hence, you should return "What is the solution".
```
Example 3:
```
Input: s = "chopper is not a tanuki", k = 5
Output: "chopper is not a tanuki"
```

Constraints:
```
1 <= s.length <= 500
```
`k` is in the range `[1, the number of words in s]`.
`s` consist of only lowercase and uppercase English letters and spaces.
The words in `s` are separated by a single space.

There are no leading or trailing spaces.

Solution:

My Solution:

```java
public class O10_TruncateSentence_1816 {
    public static void main(String[] args) {
        Solution_1816 solution_1816 = new Solution_1816();
        System.out.println(solution_11816.truncateSentence("Hello how are you Contestant", 4));
    }
}
class Solution_1816 {
    public String truncateSentence(String s, int k) {
        String[] na = s.split(" ");
        String newString = "";
        for (int i = 0; i < k; i++) {
            if (i != k - 1) {
                newString += na[i] + " ";
            } else {
                newString += na[i];
            }
        }
        return newString;
    }
}
```

// https://leetcode.com/problems/truncate-sentence/

Another Solution: 01

```java
//Approch 1:[using split]
class Solution {
    public String truncateSentence(String s, int k) {
        String[] str = s.split(" ");
        StringBuilder truncatedSentence = new StringBuilder();
        for (int i = 0; i < k - 1; i++) {
            truncatedSentence.append(str[i] + " ");
        }
        truncatedSentence.append(str[k - 1]); //because we don't want  space for last word
        return truncatedSentence.toString();
    }
}
//Complexity:
//Time:O(n) and Space:O(n)
//Note: Split function will iterate over the string to split on the basis of "". thats why time is O(n) not O(k)


//Approch 2:[without split]
class Solution {
```

```java
public String truncateSentence(String s, int k) {
    int idx = 0;
    int spaceCount = 0;

    while (idx < s.length() && spaceCount < k) {
        if (s.charAt(idx) == ' ') spaceCount++;
        idx++;
    }
    // if(spaceCount<k) means we have to include whole string
    return spaceCount == k ? s.substring(0, idx - 1) : s;
    }
}
//Complexity:
//Time:O(n) and O(1) [Big O notation average time complexity is better than approch 1]
```

Another Solution: 02

```java
class Solution {
    public String truncateSentence(String s, int k) {
        String[] words = s.split(" ");
        StringBuilder sb = new StringBuilder(words[0]);
        for (int i = 1; i < k; ++i) {
            sb.append(" " + words[i]);
        }
        return sb.toString();
    }
}
```

Another Solution: 03

```java
class Solution {
    public String truncateSentence(String s, int k) {
        int n = s.length();
        int count = 0;
        int i = 0;
        while (i < n) {
            if (s.charAt(i) == ' ') {
                count++;
                if (count == k)
                    return s.substring(0, i);
            }
            i++;
        }
        return s;
    }
}
```

09. Problem: 1365. How many numbers are smaller than the current number

Given the array `nums`, for each `nums[i]` find out how many numbers in the array are smaller than it. That is, for each `nums[i]` you have to count the number of valid `j`'s such that `j != i` and `nums[j] < nums[i]`. Return the answer in an array.

Example 1:
```
Input: nums = [8,1,2,2,3]
Output: [4,0,1,1,3]
Explanation:
For nums[0]=8 there exist four smaller numbers than it (1, 2, 2 and 3).
For nums[1]=1 does not exist any smaller number than it.
For nums[2]=2 there exist one smaller number than it (1).
For nums[3]=2 there exist one smaller number than it (1).
For nums[4]=3 there exist three smaller numbers than it (1, 2 and 2).
```
Example 2:
```
Input: nums = [6,5,4,8]
Output: [2,1,0,3]
```
Example 3:
```
Input: nums = [7,7,7,7]
Output: [0,0,0,0]
```

Constraints:
```
2 <= nums.length <= 500
0 <= nums[i] <= 100
```

Solution:

My Solution:

```java
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

public class O9_Test {
    public static void main(String[] args) {
        Solution_1365 solution_1365 = new Solution_1365();
        int[] arr = {8, 1, 2, 2, 3};
        int[] arr2 = {6, 5, 4, 8};
        int[] arr3 = {7, 7, 7, 7};
        System.out.println((Arrays.toString(solution_1365.smallerNumbersThanCurrent(arr))));
        System.out.println((Arrays.toString(solution_1365.smallerNumbersThanCurrent(arr2))));
        System.out.println((Arrays.toString(solution_1365.smallerNumbersThanCurrent(arr3))));
    }
}

class Solution_1365 {
    public int[] smallerNumbersThanCurrent(int[] nums) {
        int[] result = new int[nums.length];
        int count = 0;
        for (int i = 0; i < nums.length; i++) {
            for (int j = 0; j < nums.length; j++) {
                if (nums[i] > nums[j]) {
                    count++;
                }
```

```
            }
            result[i] = count;
            count = 0;
        }
        return result;
    }
}
```

```
/**
 * Both i and j loop start from 0.
 * so that every element is compare to every element.
 */
```

Another Solution: 01

```java
class Solution {
    public int[] smallerNumbersThanCurrent(int[] nums) {
        int[] answer = new int[nums.length];
        int count;
        for (int i = 0; i < nums.length; i++) {
            count = 0;
            for (int j = 0; j < nums.length; j++) {
                if (i != j && nums[j] < nums[i])
                    count++;
            }
            answer[i] = count;
        }
        return answer;
    }
}
```

Another Solution: 02

```java
class Solution {
    public int[] smallerNumbersThanCurrent(int[] nums) {
        int[] buck = new int[101];
        for (int i = 0; i < nums.length; i++) {
            buck[nums[i]] += 1;
        }
        for (int j = 1; j <= 100; j++) {
            buck[j] += buck[j - 1];
        }
        for (int k = 0; k < nums.length; k++) {
            int pos = nums[k];
            nums[k] = pos == 0 ? 0 : buck[pos - 1];
        }
        return nums;
    }
}
```

```
class Solution {
    public int[] smallerNumbersThanCurrent(int[] nums) {
        Map<Integer, Integer> map = new HashMap<>();
        int[] copy = nums.clone();
        Arrays.sort(copy);
        for (int i = 0; i < nums.length; i++) {
            map.putIfAbsent(copy[i], i);
        }
        for (int i = 0; i < nums.length; i++) {
            copy[i] = map.get(nums[i]);
        }
        return copy;
    }
}
```

08. Problem: 1512. Number of Good Pairs

Given an array of integers `nums`, return *the number of **good pairs***.
A pair `(i, j)` is called *good* if `nums[i] == nums[j]` and `i < j`.

Example 1:
Input: nums = [1,2,3,1,1,3]
Output: 4
Explanation: There are 4 good pairs (0,3), (0,4), (3,4), (2,5) 0-indexed.
Example 2:
Input: nums = [1,1,1,1]
Output: 6
Explanation: Each pair in the array are *good*.
Example 3:
Input: nums = [1,2,3]
Output: 0

Constraints:
```
1 <= nums.length <= 100
1 <= nums[i] <= 100
```
Accepted
362,718
Submissions
411,274

Solution:

My Solution:

```
package O1_easy.day_02;

import java.util.Arrays;
import java.util.HashMap;
```

```java
public class O8_NumberOfGoodPairs_1512 {
    public static void main(String[] args) {
        Solution_1470 solution_1470 = new Solution_1470();
        int[] arr = {1, 2, 3, 1, 1, 3};
        int[] arr2 = {1, 1, 1, 1};
        int[] arr3 = {1, 2, 3};
        System.out.println((solution_1470.numIdenticalPairs(arr)));
        System.out.println((solution_1470.numIdenticalPairs(arr2)));
        System.out.println((solution_1470.numIdenticalPairs(arr3)));
    }
}

class Solution_1470 {
    public int numIdenticalPairs(int[] nums) {
        int count = 0;
        for (int i = 0; i < nums.length; i++) {
            for (int j = 1; j < nums.length; j++) {
                if (nums[i] == nums[j] && i < j) {
                    count++;
                }
            }
        }
        return count;
    }
}
```

// https://leetcode.com/problems/number-of-good-pairs/

Another Solution: 01

```java
class Solution1 {
    public int numIdenticalPairs1(int[] A) {
        int res = 0, count[] = new int[101];
        for (int a : A) {
            res += count[a]++;
        }
        return res;
    }
}
```

Another Solution: 02

```java
class Solution2 {
    public int numIdenticalPairs2(int[] guestList) {
        HashMap<Integer, Integer> hm = new HashMap<>();
        int ans = 0;
        for (int friend : guestList) {
            int friendCount = hm.getOrDefault(friend, 0);
            ans += friendCount;
            hm.put(friend, friendCount + 1);
        }
        return ans;
```

```java
    }
}
```

```java
class Solution3 {
    public int numIdenticalPairs3(int[] nums) {
        int ans = 0;
        int[] count = new int[101];
        for (int n : nums)
            count[n]++;
        for (int n : count)
            ans += (n * (n - 1)) / 2;
        return ans;
    }
}
```

07. Problem: 2011. Final value of variable after performing operation

There is a programming language with only **four** operations and **one** variable x:

++X and X++ **increments** the value of the variable x by 1.

--X and X-- **decrements** the value of the variable x by 1.

Initially, the value of x is 0.

Given an array of strings operations containing a list of operations, return *the **final** value of x after performing all the operations*.

Example 1:
```
Input: operations = ["--X","X++","X++"]
Output: 1
Explanation: The operations are performed as follows:
Initially, X = 0.
--X: X is decremented by 1, X =  0 - 1 = -1.
X++: X is incremented by 1, X = -1 + 1 =  0.
X++: X is incremented by 1, X =  0 + 1 =  1.
```
Example 2:
```
Input: operations = ["++X","++X","X++"]
Output: 3
Explanation: The operations are performed as follows:
Initially, X = 0.
++X: X is incremented by 1, X = 0 + 1 = 1.
++X: X is incremented by 1, X = 1 + 1 = 2.
X++: X is incremented by 1, X = 2 + 1 = 3.
```
Example 3:
```
Input: operations = ["X++","++X","--X","X--"]
Output: 0
Explanation: The operations are performed as follows:
Initially, X = 0.
X++: X is incremented by 1, X = 0 + 1 = 1.
++X: X is incremented by 1, X = 1 + 1 = 2.
--X: X is decremented by 1, X = 2 - 1 = 1.
X--: X is decremented by 1, X = 1 - 1 = 0.
```

Constraints:

```
1 <= operations.length <= 100
```
`operations[i]` will be either `"++X"`, `"X++"`, `"--X"`, or `"X--"`.

Solution:

My Solution:

```java
import java.util.Arrays;

public class O7_Test {
    public static void main(String[] args) {
        Solution_2011 solution_2011 = new Solution_2011();
        String[] arr = {"--X", "X++", "X++"};
        String[] arr2 = {"X++", "++X", "--X", "X--"};
        String[] arr3 = {"++X", "++X", "X++"};
        System.out.println(solution_2011.finalValueAfterOperations(arr));
        System.out.println(solution_2011.finalValueAfterOperations(arr2));
        System.out.println(solution_2011.finalValueAfterOperations(arr3));


    }
}

class Solution_2011 {
    public int finalValueAfterOperations(String[] operations) {

        int result = 0;
        for (int i = 0; i < operations.length; i++) {
            if (operations[i].equals("++X")) {
                result = 1 + result;
            } else if (operations[i].equals("--X")) {
                result = result - 1;
            } else if (operations[i].equals("X++")) {
                result = result + 1;
            } else if (operations[i].equals("X--")) {
                result = result - 1;
            }
        }
        return result;
    }
}
```

// https://leetcode.com/problems/build-array-from-permutation/

Another Solution: 01

```java
class Solution1 {
    public int finalValueAfterOperations1(String[] operations) {
        int val = 0;
        for (int i = 0; i < operations.length; i++) {
```

```
            if (operations[i].charAt(1) == '+') val++;
            else val--;
        }
        return val;
    }
}
```

```
class Solution2 {
    public int finalValueAfterOperations2(String[] operations) {
        int res = 0;
        for (String operation : operations) {
            res += operation.charAt(1) == '+' ? 1 : -1;
        }
        return res;
    }
}
```

```
    public static int finalValueAfterOperations3(String[] operations) {
        return Arrays.stream(operations, 0, operations.length)
            .mapToInt(operation -> operation.charAt(1) == '+' ? 1 : -1)
            .sum();
    }
```

06. Problem: 1929. Concatenation of Array

Given an integer array `nums` of length `n`, you want to create an array `ans` of length `2n` where `ans[i] == nums[i]` and `ans[i + n] == nums[i]` for `0 <= i < n` (**0-indexed**).

Specifically, `ans` is the **concatenation** of two `nums` arrays.

Return *the array* `ans`.

Example 1:
```
Input: nums = [1,2,1]
Output: [1,2,1,1,2,1]
Explanation: The array ans is formed as follows:
- ans = [nums[0],nums[1],nums[2],nums[0],nums[1],nums[2]]
- ans = [1,2,1,1,2,1]
```
Example 2:
```
Input: nums = [1,3,2,1]
Output: [1,3,2,1,1,3,2,1]
Explanation: The array ans is formed as follows:
- ans = [nums[0],nums[1],nums[2],nums[3],nums[0],nums[1],nums[2],nums[3]]
- ans = [1,3,2,1,1,3,2,1]
```

Constraints:
```
n == nums.length
1 <= n <= 1000
1 <= nums[i] <= 1000
```

Solution:

My Solution:

```java
public class O6_ConcatenationOfArray_1929 {
    public static void main(String[] args) {
        Solution_1929 solution_1929 = new Solution_1929();
        int[] arr = {1, 3, 2, 1};
        System.out.println(Arrays.toString(solution_1929.getConcatenation(arr)));
    }
}
class Solution_1929 {
    public int[] getConcatenation(int[] nums) {
        int[] newAnswer = new int[nums.length * 2];
        int givenArrayLength = nums.length;
        for (int i = 0; i < nums.length; i++) {
            newAnswer[i] = nums[i];
            newAnswer[i + givenArrayLength] = nums[i];
        }
        return newAnswer;
    }
}
```

// https://leetcode.com/problems/concatenation-of-array/

```
/**
 * (int i = 0; i < nums.length; i++) --> must be i<nums.length or i<=nums.length-1
 */
```

Another Solution: 01

```java
public int[] getConcatenation(int[] nums) {
    int[] result = new int[nums.length * 2];
    for (int i = 0; i < nums.length; i++)
        result[i + nums.length] = result[i] = nums[i];
    return result;
}
```

Another Solution: 02

```java
public int[] getConcatenation(int[] nums) {
    int[] ans = new int[nums.length * 2];
    for (int i = 0; i < nums.length; i++) {
        ans[i] = nums[i];
        ans[nums.length + i] = nums[i];
    }
    return ans;
}
```

5. Problem: 1108. Defanging an IP Address

Given a valid (IPv4) IP `address`, return a defanged version of that IP address.
A *defanged IP address* replaces every period "`.`" with "`[.]`".

Example 1:
```
Input: address = "1.1.1.1"
Output: "1[.]1[.]1[.]1"
```
Example 2:
```
Input: address = "255.100.50.0"
Output: "255[.]100[.]50[.]0"
```

Constraints:
The given `address` is a valid IPv4 address.

Solution:

My Solution:

```java
public class O5_DefiningIpAddress_1108 {
    public static void main(String[] args) {
        Solution_1108 solution_1108 = new Solution_1108();
        System.out.println(solution_1108.defangIPaddr("1.1.1.1"));
    }
}

class Solution_1108 {
    public String defangIPaddr(String address) {
        return address.replace(".", "[.]");
    }
}
```

Another Solution: 01

```java
// Another solution:
class Solution {
    public String defangIPaddr(String address) {
        StringBuilder str = new StringBuilder();
        for (int i = 0; i < address.length(); i++) {
            if (address.charAt(i) == '.') {
                str.append("[.]");
            } else {
                str.append(address.charAt(i));
            }
        }
    }
```

```java
        return str.toString();
    }
}
```

```java
//Another Solution:
class Solution {
    public String defangIPaddr(String address) {
        return address.replaceAll("\\.", "[.]");
    }
}
```

```java
String str = "";
for(int itr = 0; itr < address.length(); itr++){
    if(address.charAt(itr) == '.') str += "[.]";
    else str += address.charAt(itr);
}
return str;
```

04. Problem: 1662. Check If Two String array are Equivalent

Given two string arrays `word1` and `word2`, return `true` if the two arrays **represent** the same string, and `false` otherwise.

A string is **represented** by an array if the array elements concatenated **in order** forms the string.

Example 1:
```
Input: word1 = ["ab", "c"], word2 = ["a", "bc"]
Output: true
Explanation:
word1 represents string "ab" + "c" -> "abc"
word2 represents string "a" + "bc" -> "abc"
The strings are the same, so return true.
```
Example 2:
```
Input: word1 = ["a", "cb"], word2 = ["ab", "c"]
Output: false
```
Example 3:
```
Input: word1  = ["abc", "d", "defg"], word2 = ["abcddefg"]
Output: true
```

Constraints:
```
1 <= word1.length, word2.length <= 10³
1 <= word1[i].length, word2[i].length <= 10³
1 <= sum(word1[i].length), sum(word2[i].length) <= 10³
```
`word1[i]` and `word2[i]` consist of lowercase letters.

Solution:

```java
public class O4_StringEquivalent_1662 {
    public static void main(String[] args) {
        Solution_1662 solution_1662 = new Solution_1662();
        String[] word1 = {"a", "cb"};
        String[] word2 = {"ab", "c"};
        System.out.println(solution_1662.arrayStringsAreEqual(word1, word2));
    }
}

class Solution_1662 {
    public boolean arrayStringsAreEqual(String[] word1, String[] word2) {
        String r1 = "";
        for (int i = 0; i < word1.length; i++) {
            r1 += word1[i];
        }
        System.out.println(r1);
        String r2 = "";
        for (int i = 0; i < word2.length; i++) {
            r2 += word2[i];
        }
        System.out.println(r2);

        if (r1.equals(r2)) {
            return true;
        } else {
            return false;
        }
    }
}
```

// https://leetcode.com/problems/check-if-two-string-arrays-are-equivalent/

```java
class Solution {
    public boolean arrayStringsAreEqual(String[] word1, String[] word2) {
        return String.join("", word1).equals(String.join("", word2));
    }
}
```

```java
class Solution {
    public boolean arrayStringsAreEqual(String[] word1, String[] word2) {
        StringBuilder sb1 = new StringBuilder(), sb2 = new StringBuilder();
        for (String word : word1)
            sb1.append(word);

        for (String word : word2)
```

```
            sb2.append(word);

        return sb1.toString().equals(sb2.toString());
    }
}
```

```
class Solution {
    public boolean arrayStringsAreEqual(String[] word1, String[] word2) {
        int idx1 = 0, idx2 = 0, arrIdx1 = 0, arrIdx2 = 0;
        while (arrIdx1 < word1.length && arrIdx2 < word2.length) {
            if (word1[arrIdx1].charAt(idx1) != word2[arrIdx2].charAt(idx2)) return false;
            if (idx1 == word1[arrIdx1].length() - 1) {
                idx1 = 0;
                arrIdx1++;
            } else idx1++;
            if (idx2 == word2[arrIdx2].length() - 1) {
                idx2 = 0;
                arrIdx2++;
            } else idx2++;
        }
        return arrIdx1 == word1.length && arrIdx2 == word2.length;
    }
}
```

```
class Solution {
    public boolean arrayStringsAreEqual(String[] word1, String[] word2) {
        int idx1 = 0, idx2 = 0, arrIdx1 = 0, arrIdx2 = 0;
        while (arrIdx1 < word1.length && arrIdx2 < word2.length) {
            if (word1[arrIdx1].charAt(idx1) != word2[arrIdx2].charAt(idx2)) return false;
            idx1 = (++idx1) % word1[arrIdx1].length();
            idx2 = (++idx2) % word2[arrIdx2].length();
            if (idx1 == 0) arrIdx1++;
            if (idx2 == 0) arrIdx2++;
        }
        return arrIdx1 == word1.length && arrIdx2 == word2.length;
    }
}
```

03. Problem: 1480. Running Sum of 1d array

Given an array `nums`. We define a running sum of an array as `runningSum[i] = sum(nums[0]…nums[i])`.
Return the running sum of `nums`.

Example 1:
```
Input: nums = [1,2,3,4]
Output: [1,3,6,10]
```

Explanation: Running sum is obtained as follows: [1, 1+2, 1+2+3, 1+2+3+4].
Example 2:
Input: nums = [1,1,1,1,1]
Output: [1,2,3,4,5]
Explanation: Running sum is obtained as follows: [1, 1+1, 1+1+1, 1+1+1+1, 1+1+1+1+1].
Example 3:
Input: nums = [3,1,2,10,1]
Output: [3,4,6,16,17]

Constraints:
```
1 <= nums.length <= 1000
-10^6 <= nums[i] <= 10^6
```

Solution:

My Solution:

```java
import java.util.Arrays;

public class O3_RunningSum1dArray_1480 {
    public static void main(String[] args) {
        Solution_1480 solution_1480 = new Solution_1480();
        int[] nums = {1, 1, 1, 1, 1};
        System.out.println(Arrays.toString(solution_1480.runningSum(nums)));
        int[] nums2 = {1, 2, 3, 4};
        System.out.println(Arrays.toString(solution_1480.runningSum(nums2)));
    }
}

class Solution_1480 {
    public int[] runningSum(int[] nums) {
        int[] result = new int[nums.length];
        int sum = 0;
        for (int i = 0; i < nums.length; i++) {
            sum = sum + nums[i];
            result[i] = sum;
        }
        return result;
    }
}

    /**
     * {1, 2, 3, 4}
     * <p>
     * Iteration: 1
     * sum = sum + num[i]  | sum = 0 + 1 = 1
     * result[i] = sum     | result[0] = 1
     * <p>
     * Iteration: 2
     * sum = 1 + 2 = 3 , result[1] = 3
     * <p>
```

```
     * Iteration: 3
     * sum = 3 + 3 = 6 , result[2] = 6
     * <p>
     * Iteration: 4
     * sum = 3 + 6 = 10 , result[3] = 10
     */
```

// https://leetcode.com/problems/running-sum-of-1d-array/

Another Solution: 01

```java
public int[] runningSum(int[] nums) {
    // modify the input array, adding n[i] with n[i-1]
    for (int i = 1; i < nums.length; i++) {
        nums[i] += nums[i - 1];
    }

    // return the modified array
    return nums;
}
```

Another Solution: 02

```java
class Solution {
    public int[] runningSum(int[] nums) {
        return IntStream.range(0, nums.length).map(i -> i == 0 ? nums[i] : (nums[i] += nums[i - 1])).toArray();
    }
}
```

Another Solution: 03

02. Problem: 9. Palindrome or not

Given an integer `x`, return `true` if `x` is palindrome integer.
An integer is a **palindrome** when it reads the same backward as forward.
For example, `121` is a palindrome while `123` is not.

Example 1:
Input: x = 121
Output: true
Explanation: 121 reads as 121 from left to right and from right to left.
Example 2:
Input: x = -121
Output: false
Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.
Example 3:
Input: x = 10
Output: false
Explanation: Reads 01 from right to left. Therefore it is not a palindrome.

Constraints:

-2^{31} <= x <= 2^{31} - 1

Follow up: Could you solve it without converting the integer to a string?

Solution:

My Solution:

```java
package O1_easy;

public class O2_Palindrome_9 {
    public static void main(String args[]) {
        Solution_9 solution = new Solution_9();
        System.out.println(solution.isPalindrome(121));
        System.out.println(solution.isPalindrome(123));
    }
}

class Solution_9 {
    public boolean isPalindrome(int x) {
        String s = String.valueOf(x);
        StringBuffer sb = new StringBuffer(s);
        String s1 = String.valueOf(sb.reverse());

        if (s.equals(s1)) {
            return true;
        } else {
            return false;
        }
    }
}

// https://leetcode.com/problems/palindrome-number/
```

Another Solution: 01

```java
public class Solution {
    public boolean isPalindrome(int x) {
        if (x < 0) return false;
        int y = x;
        int res = 0;
        while (y != 0) {
            res = res * 10 + y % 10;
            y /= 10;
        }
```

```java
        return x == res;
    }
}
```

```java
public boolean isPalindrome(int x){
    int p=x,q=0;
    while(p>=1){
    q*=10;
    q+=p%10;
    p/=10;
    }
    return q==x;
    }
```

01. Problem: 2235. Add Two Integers

Given two integers num1 and num2, return the sum of the two integers.

Example 1:
```
Input: num1 = 12, num2 = 5
Output: 17
Explanation: num1 is 12, num2 is 5, and their sum is 12 + 5 = 17, so 17 is returned.
```
Example 2:
```
Input: num1 = -10, num2 = 4
Output: -6
Explanation: num1 + num2 = -6, so -6 is returned.
```

Constraints:
```
-100 <= num1, num2 <= 100
```

Solution:

My Solution:

```java
public class O1_AddTwoIntegers_2235 {
    public static void main(String[] args) {
        Solution s = new Solution();
        System.out.println(s.sum(5, 8));
    }
}

class Solution {
    public int sum(int num1, int num2) {
        return num1 + num2;
```

```
    }
}
```

//

Another Solution: 01

```
class Solution {
    public int sum(int num1, int num2) {
        if (num2 == 0) return num1;
        int temp = (num1 & num2) << 1;
        return sum(num1 ^ num2, temp);
    }
}
```

Another Solution: 02

Another Solution: 03

Day 00 -1929,,,,,.

5. Problem:

Solution:

My Solution:

Another Solution: 01

Another Solution: 02

Another Solution: 03

04. Problem:

Solution:

My Solution:

Another Solution: 01

Another Solution: 02

Another Solution: 03

03. Problem:

Solution:

My Solution:

Another Solution: 01

Another Solution: 02

Another Solution: 03

02. Problem:

Solution:

My Solution:

Another Solution: 01

Another Solution: 02

Another Solution: 03

01. Problem:

Solution:

My Solution:

Another Solution: 01

Another Solution: 02

Another Solution: 03

06. Problem: 1929. Concatenation of Array

Solution:
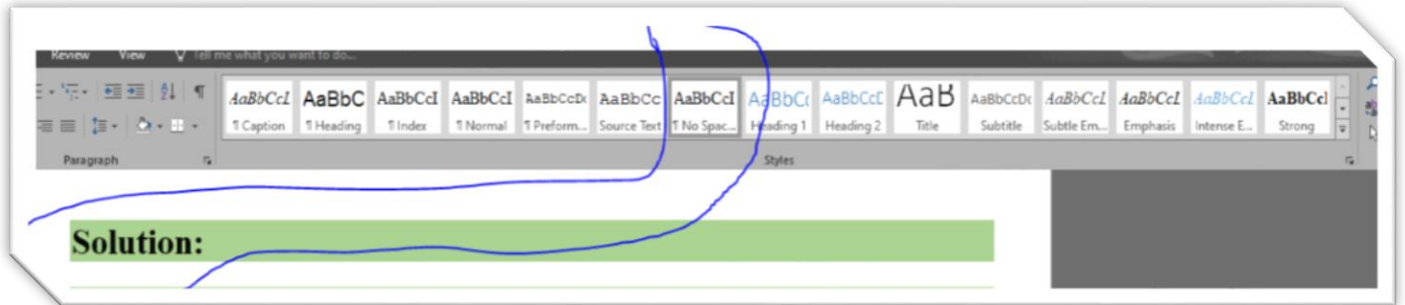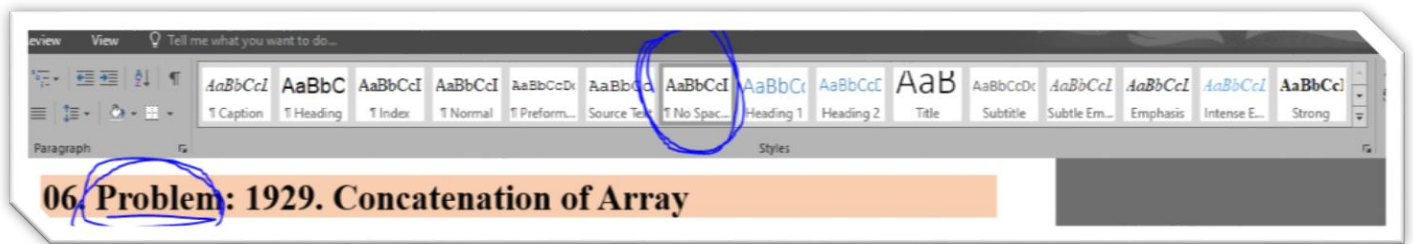
```
A a = new A(); a.run(); // A
A a2 = new B(); a2.run(); // B
B a3 = new B(); a3.run(); // B

//B a4 = new A(); a3.run(); // Not Allow, compile time error
B a5 = (B) new A(); a3.run(); // Not Allow, run time error, cannot cast
```