First Bad Version

You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad. Suppose you have n versions [1, 2, ..., n] and you want to find out the first bad one, which causes all the following ones to be bad.
You are given an API bool isBadVersion(version) which returns whether version is bad. Implement a function to find the first bad version. You should minimize the number of calls to the API.

Example 1:
Input: n = 5, bad = 4
Output: 4
Explanation:
call isBadVersion(3) -> false
call isBadVersion(5) -> true
call isBadVersion(4) -> true
Then 4 is the first bad version.
Example 2:
Input: n = 1, bad = 1
Output: 1

Power of Four

Given an integer n, return *true if it is a power of four. Otherwise, return false*. An integer n is a power of four, if there exists an integer x such that n == 4_x.

Example 1:
Input: n = 16
Output: true

Example 2:
Input: n = 5
Output: false

Example 3:
Input: n = 1
Output: true

Find the Difference of Two Arrays

Given two **0-indexed** integer arrays nums1 and nums2, return *a list* answer *of size* 2 *where:* ● answer[0] *is a list of all* ***distinct*** *integers in* nums1 *which are* ***not*** *present in* nums2*.* ● answer[1] *is a list of all* ***distinct*** *integers in* nums2 *which are* ***not*** *present in* nums1. **Note** that the integers in the lists may be returned in **any** order.

Example 1:
Input: nums1 = [1,2,3], nums2 = [2,4,6]
Output: [[1,3],[4,6]]
Explanation:
For nums1, nums1[1] = 2 is present at index 0 of nums2, whereas nums1[0] = 1 and nums1[2] =

3 are not present in nums2. Therefore, answer[0] = [1,3].
For nums2, nums2[0] = 2 is present at index 1 of nums1, whereas nums2[1] = 4 and nums2[2] = 6 are not present in nums2. Therefore, answer[1] = [4,6].
Example 2:
Input: nums1 = [1,2,3,3], nums2 = [1,1,2,2]
Output: [[3],[]]
Explanation:
For nums1, nums1[2] and nums1[3] are not present in nums2. Since nums1[2] == nums1[3], their value is only included once and answer[0] = [3].
Every integer in nums2 is present in nums1. Therefore, answer[1] = [].

Intersection of Two Arrays II

Given two integer arrays nums1 and nums2, return *an array of their intersection*. Each element in the result must appear as many times as it shows in both arrays and you may return the result in **any order**.
Example 1:
Input: nums1 = [1,2,2,1], nums2 = [2,2]
Output: [2,2]
Example 2:
Input: nums1 = [4,9,5], nums2 = [9,4,9,8,4]
Output: [4,9]
Explanation: [9,4] is also accepted.

Number of Unequal Triplets in Array

You are given a **0-indexed** array of positive integers nums. Find the number of triplets (i, j, k) that meet the following conditions:

- 0 <= i < j < k < nums.length
- nums[i], nums[j], and nums[k] are **pairwise distinct**.
- In other words, nums[i] != nums[j], nums[i] != nums[k], and nums[j] != nums[k].

Return *the number of triplets that meet the conditions.*
Example 1:
Input: nums = [4,4,2,4,3]
Output: 3
Explanation: The following triplets meet the conditions:
- (0, 2, 4) because 4 != 2 != 3
- (1, 2, 4) because 4 != 2 != 3
- (2, 3, 4) because 2 != 4 != 3
Since there are 3 triplets, we return 3.
Note that (2, 0, 4) is not a valid triplet because 2 > 0.

Example 2:
Input: nums = [1,1,1,1,1]
Output: 0
Explanation: No triplets meet the conditions so we return 0.

Check if Number is a Sum of Powers of Three

Given an integer n, return true *if it is possible to represent* n *as the sum of distinct powers of three.* Otherwise, return false.
An integer y is a power of three if there exists an integer x such that y == 3_x.

Example 1:
Input: n = 12
Output: true
Explanation: 12 = 3_1 + 3_2

Example 2:
Input: n = 91
Output: true
Explanation: 91 = 3_0 + 3_2 + 3_4

Example 3:
Input: n = 21
Output: false

Number of 1 Bits

Write a function that takes the binary representation of an unsigned integer and returns the number of '1' bits it has (also known as the Hamming weight).
Note:

- Note that in some languages, such as Java, there is no unsigned integer type. In this case, the input will be given as a signed integer type. It should not affect your implementation, as the integer's internal binary representation is the same, whether it is signed or unsigned.
- In Java, the compiler represents the signed integers using 2's complement notation. Therefore, in **Example 3**, the input represents the signed integer. -3.

Example 1:
Input: n = 00000000000000000000000000001011
Output: 3
Explanation: The input binary string **00000000000000000000000000001011** has a total of three '1' bits.

Example 2:
Input: n = 00000000000000000000000010000000
Output: 1
Explanation: The input binary string **00000000000000000000000010000000** has a total of one '1' bit.

Example 3:
Input: n = 11111111111111111111111111111101
Output: 31
Explanation: The input binary string **11111111111111111111111111111101** has a total of thirty one '1' bits.

Constraints:

- The input must be a **binary string** of length 32.

Find the leaders in the array

Input:

n = 6
A[] = {16,17,4,3,5,2}

Output: 17 5 2
Explanation: The first leader is 17 as it is greater than all the elements to its right. Similarly, the next leader is 5. The right most element is always a leader so it is also included.
Example 2:

Input:
n = 5
A[] = {1,2,3,4,0}
Output: 4 0

Reduce the size of this string using mathematical logic

Capgemini in its online written test has a coding question, wherein the students are given a string with multiple characters that are repeated consecutively. You're supposed to reduce the size of this string using mathematical logic given as in the example below:

Ex1:
Input :
aabbbbeeeeffggg

Output:
a2b4e4f2g3

Ex2:
Input :
abbccccc

Output:
ab2c5

Given a string s, find the length of the longest substring without repeating characters.

Example 1:

Input: s = "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.
Example 2:

Input: s = "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.
Example 3:

Input: s = "pwwkew"
Output: 3
Explanation: The answer is "wke", with the length of 3.
Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

Example 1:
Input: s = "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.

Example 2:
Input: s = "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.

Example 3:
Input: s = "pwwkew"
Output: 3
Explanation: The answer is "wke", with the length of 3.
Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

Valid Palindrome

A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.
Given a string s, return true *if it is a **palindrome**, or* false *otherwise*.

Example 1:
Input: s = "A man, a plan, a canal: Panama"
Output: true
Explanation: "amanaplanacanalpanama" is a palindrome.

Example 2:
Input: s = "race a car"
Output: false
Explanation: "raceacar" is not a palindrome.

Example 3:
Input: s = " "
Output: true
Explanation: s is an empty string "" after removing non-alphanumeric characters. Since an empty string reads the same forward and backward, it is a palindrome.

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

Contains Duplicate II

Given an integer array nums and an integer k, return true *if there are two* ***distinct indices*** i *and* j *in the array such that* nums[i] == nums[j] *and* abs(i - j) <= k.

Example 1:
Input: nums = [1,2,3,1], k = 3
Output: true

Example 2:
Input: nums = [1,0,1,1], k = 1
Output: true

Example 3:
Input: nums = [1,2,3,1,2,3], k = 2
Output: false

Reverse String

Write a function that reverses a string. The input string is given as an array of characters s. You must do this by modifying the input array in-place with O(1) extra memory.

Example 1:
Input: s = ["h","e","l","l","o"]
Output: ["o","l","l","e","h"]

Example 2:
Input: s = ["H","a","n","n","a","h"]
Output: ["h","a","n","n","a","H"]

Valid Anagram

Given two strings s and t, return true *if* t *is an anagram of* s*, and* false *otherwise*. An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Example 1:
Input: s = "anagram", t = "nagaram"
Output: true

Example 2:
Input: s = "rat", t = "car"
Output: false

Missing Number

Given an array nums containing n distinct numbers in the range [0, n], return *the only number in the range that is missing from the array.*

Example 1:
Input: nums = [3,0,1]
Output: 2
Explanation: n = 3 since there are 3 numbers, so all numbers are in the range [0,3]. 2 is the missing number in the range since it does not appear in nums.

Example 2:
Input: nums = [0,1]
Output: 2
Explanation: n = 2 since there are 2 numbers, so all numbers are in the range [0,2]. 2 is the missing number in the range since it does not appear in nums.

Example 3:
Input: nums = [9,6,4,2,3,5,7,0,1]
Output: 8
Explanation: n = 9 since there are 9 numbers, so all numbers are in the range [0,9]. 8 is the missing number in the range since it does not appear in nums.

Excel Sheet Column Number

Given a string columnTitle that represents the column title as appears in an Excel sheet, return *its corresponding column number.*
For example:
A -> 1
B -> 2
C -> 3
...
Z -> 26
AA -> 27
AB -> 28
...

Example 1:
Input: columnTitle = "A"
Output: 1

Example 2:

Input: columnTitle = "AB"
Output: 28

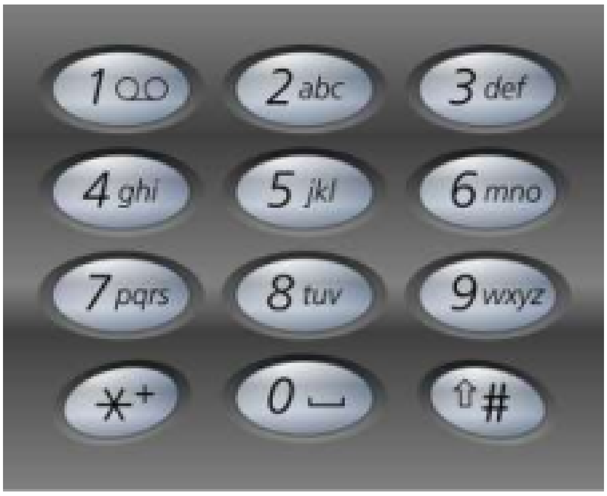
Example 3:
Input: columnTitle = "ZY"
Output: 701

```
    base26Value = c -      +  ;
    power = columnTitle.length() -   - i;
result += base26Value * Math.pow(  , power);
```

Letter Combinations of a Phone Number

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.



Example 1:
Input: digits = "23"
Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]

Example 2:
Input: digits = ""
Output: []

Example 3:
Input: digits = "2"
Output: ["a","b","c"]

Climbing Stairs

You are climbing a staircase. It takes n steps to reach the top.
Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

Example 1:
Input: n = 2
Output: 2
Explanation: There are two ways to climb to the top.
1. 1 step + 1 step
2. 2 steps

Example 2:
Input: n = 3
Output: 3
Explanation: There are three ways to climb to the top.
1. 1 step + 1 step + 1 step
2. 1 step + 2 steps
3. 2 steps + 1 step

```
if (n < 2) {
 return n; // base cases
 }

 int[] dp = new int[n + 1];
 dp[0] = 1;
 dp[1] = 1;

 for (int i = 2; i <= n; i++) {
 dp[i] = dp[i - 1] + dp[i - 2];
 }
 return dp[n];
```

Min Cost Climbing Stairs

You are given an integer array cost where cost[i] is the cost of i_{th} step on a staircase. Once you pay the cost, you can either climb one or two steps.
You can either start from the step with index 0, or the step with index 1.
Return *the minimum cost to reach the top of the floor*.

Example 1:
Input: cost = [10,15,20]
Output: 15
Explanation: You will start at index 1.
- Pay 15 and climb two steps to reach the top.
The total cost is 15.

Example 2:
Input: cost = [1,100,1,1,1,100,1,1,100,1]
Output: 6
Explanation: You will start at index 0.
- Pay 1 and climb two steps to reach index 2.
- Pay 1 and climb two steps to reach index 4.
- Pay 1 and climb two steps to reach index 6.
- Pay 1 and climb one step to reach index 7.
- Pay 1 and climb two steps to reach index 9.
- Pay 1 and climb one step to reach the top.
The total cost is 6.