Unit 2: Control Structure, Operators, and Functions

1. **Merge Two Sorted Lists (#21)**

You are given the heads of two sorted linked lists list1 and list2. Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.
Return *the head of the merged linked list*.
Example 1:
Input: list1 = [1,2,4], list2 = [1,3,4]
Output: [1,1,2,3,4,4]
Example 2:
Input: list1 = [], list2 = []
Output: []
Example 3:
Input: list1 = [], list2 = [0]
Output: [0]

2. **Reverse Linked List (#206)**

Given the head of a singly linked list, reverse the list, and return *the reversed list*.
Example 1:
Input: head = [1,2,3,4,5]
Output: [5,4,3,2,1]
Example 2:
Input: head = [1,2]
Output: [2,1]
Example 3:
Input: head = []
Output: []

3. **Path Sum (#112)**

Given the root of a binary tree and an integer targetSum, return true if the tree has a **root-to-leaf** path such that adding up all the values along the path equals targetSum. A **leaf** is a node with no children.
Example 1:
Input: root = [5,4,8,11,null,13,4,7,2,null,null,null,1], targetSum = 22
Output: true
Explanation: The root-to-leaf path with the target sum is shown.
Example 2:
Input: root = [1,2,3], targetSum = 5
Output: false
Explanation: There are two root-to-leaf paths in the tree:
(1 --> 2): The sum is 3.
(1 --> 3): The sum is 4.
There is no root-to-leaf path with sum = 5.
Example 3:
Input: root = [], targetSum = 0
Output: false
Explanation: Since the tree is empty, there are no root-to-leaf paths.

4. **Linked List Cycle**

Given head, the head of a linked list, determine if the linked list has a cycle in it. There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. **Note that pos is not passed as a parameter**. Return true *if there is a cycle in the linked list*. Otherwise, return false.

Example 1:
Input: head = [3,2,0,-4], pos = 1
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).
Example 2:
Input: head = [1,2], pos = 0
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 0th node.
Example 3:
Input: head = [1], pos = -1
Output: false
Explanation: There is no cycle in the linked list.

5. **First Bad Version (#278)**

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

6. **Add Digits (#258)**

Given an integer num, repeatedly add all its digits until the result has only one digit, and return it.
Example 1:
Input: num = 38
Output: 2
Explanation: The process is
38 --> 3 + 8 --> 11
11 --> 1 + 1 --> 2
Since 2 has only one digit, return it.
Example 2:
Input: num = 0
Output: 0

7. **Climbing Stairs (#70)**

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

8. **Remove Duplicates from Sorted Array (#26)**

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. Then return *the number of unique elements in* nums.
Consider the number of unique elements of nums to be k, to get accepted, you need to do the following things:

- Change the array nums such that the first k elements of nums contain the unique elements in the order they were present in nums initially. The remaining elements of nums are not important as well as the size of nums.
- Return k.

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).

9. **Length of Last Word (#58)**

Given a string s consisting of words and spaces, return *the length of the* ***last*** *word in the string.*
A **word** is a maximal substring consisting of non-space characters only.
Example 1:
Input: s = "Hello World"
Output: 5
Explanation: The last word is "World" with length 5.
Example 2:
Input: s = "   fly me   to   the moon "
Output: 4
Explanation: The last word is "moon" with length 4.
Example 3:
Input: s = "luffy is still joyboy"
Output: 6
Explanation: The last word is "joyboy" with length 6.

10. **Maximum Subarray (#53)**

Given an integer array nums, find the subarray with the largest sum, and return *its sum*.
Example 1:
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: The subarray [4,-1,2,1] has the largest sum 6.
Example 2:
Input: nums = [1]
Output: 1
Explanation: The subarray [1] has the largest sum 1.
Example 3:
Input: nums = [5,4,-1,7,8]
Output: 23
Explanation: The subarray [5,4,-1,7,8] has the largest sum 23.

11. **Best Time to Buy and Sell Stock II (#122)**

You are given an integer array prices where prices[i] is the price of a given stock on the ith day. On each day, you may decide to buy and/or sell the stock. You can only hold **at most one** share of the stock at any time. However, you can buy it then immediately sell it on the **same day**. Find and return *the* ***maximum*** *profit you can achieve*.
Example 1:
Input: prices = [7,1,5,3,6,4]
Output: 7
Explanation: Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit = 5-1 = 4.
Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3.
Total profit is 4 + 3 = 7.
Example 2:
Input: prices = [1,2,3,4,5]

Output: 4
Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4.
Total profit is 4.
Example 3:
Input: prices = [7,6,4,3,1]
Output: 0
Explanation: There is no way to make a positive profit, so we never buy the stock to achieve the maximum profit of 0.

12. **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.

13. **Longest Common Prefix (#14)**

Write a function to find the longest common prefix string amongst an array of strings. If there is no common prefix, return an empty string "".
Example 1:
Input: strs = ["flower","flow","flight"]
Output: "fl"
Example 2:
Input: strs = ["dog","racecar","car"]
Output: ""
Explanation: There is no common prefix among the input strings.

14. **Counting Bits (#338)**

Given an integer n, return *an array* ans *of length* n + 1 *such that for each* i (0 <= i <= n)*,* ans[i] *is the number of* 1*'s in the binary representation of* i.
Example 1:
Input: n = 2
Output: [0,1,1]
Explanation:
0 --> 0
1 --> 1
2 --> 10

Example 2:
Input: n = 5
Output: [0,1,1,2,1,2]
Explanation:
0 --> 0
1 --> 1
2 --> 10
3 --> 11
4 --> 100
5 --> 101

15. **Rotate Array (#189)**

Given an integer array nums, rotate the array to the right by k steps, where k is non-negative.
Example 1:
Input: nums = [1,2,3,4,5,6,7], k = 3
Output: [5,6,7,1,2,3,4]
Explanation:
rotate 1 steps to the right: [7,1,2,3,4,5,6]
rotate 2 steps to the right: [6,7,1,2,3,4,5]
rotate 3 steps to the right: [5,6,7,1,2,3,4]
Example 2:
Input: nums = [-1,-100,3,99], k = 2
Output: [3,99,-1,-100]
Explanation:
rotate 1 steps to the right: [99,-1,-100,3]
rotate 2 steps to the right: [3,99,-1,-100]

16. **Length of Last Word (#58)**

Given a string s consisting of words and spaces, return *the length of the* ***last*** *word in the string.* A **word** is a maximal substring consisting of non-space characters only.
Example 1:
Input: s = "Hello World"
Output: 5
Explanation: The last word is "World" with length 5.
Example 2:
Input: s = "   fly me   to   the moon "
Output: 4
Explanation: The last word is "moon" with length 4.
Example 3:
Input: s = "luffy is still joyboy"
Output: 6
Explanation: The last word is "joyboy" with length 6.

17. **Reverse Bits (#190)**

Reverse bits of a given 32 bits unsigned integer.

Example 1:

Input: n = 43261596

Output: 964176192

Explanation:

Integer	Binary
43261596	00000010100101000001111010011100
964176192	00111001011110000010100101000000

Example 2:

Input: n = 2147483644

Output: 1073741822

Explanation:

Integer	Binary
2147483644	01111111111111111111111111111100
1073741822	00111111111111111111111111111110

18. **Reverse String (#344)**

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"]

19. **Majority Element (#169)**

Given an array nums of size n, return *the majority element*. The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array.

Example 1:

Input: nums = [3,2,3]

Output: 3

Example 2:

Input: nums = [2,2,1,1,1,2,2]

Output: 2

20. **Best Time to Buy and Sell Stock (#121)**

You are given an array prices where prices[i] is the price of a given stock on the i^{th} day. You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock. Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return 0.

Example 1:

Input: prices = [7,1,5,3,6,4]

Output: 5

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.
Example 2:
Input: prices = [7,6,4,3,1]
Output: 0
Explanation: In this case, no transactions are done and the max profit = 0.