**Day 3**

**Function $ RECURSION DSA QUESTION**

**Very Easy**

Q1 .**Fibonnacci Series Using Recursion**

The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is,

F(0) = 0, F(1) = 1

F(n) = F(n - 1) + F(n - 2), for n > 1.

Given n, calculate F(n).

**Example 1:**

Input: n = 2

Output: 1

Explanation: F(2) = F(1) + F(0) = 1 + 0 = 1

**Example 2:**

Input: n = 3

Output: 2

Explanation: F(3) = F(2) + F(1) = 1 + 1 = 2.

**Constraints:**

0 <= n <= 30

**Q2 . Factorial Of Number Using Recursion**

Write a program that returns the value of N! (N factorial) using recursion.  
Note that N! =- 1\*2\*...\*N  
Also, 0! = 1 and 1! = 1.

**Input Format**

One number, N.

**Constraints**

0 <= N <= 15

**Output Format**

Output the value of N factorial.

**Q3.Sum of Natural Number Using Recursion**

Given a number n, find sum of first n natural numbers. To calculate the sum, we will use a recursive function recur\_sum().

**Examples 1:**

Input : 3

Output : 6

Explanation : 1 + 2 + 3 = 6

**Examples 2:**

Input : 5

Output : 15

Explanation : 1 + 2 + 3 + 4 + 5 = 15

Below is code to find the sum of natural numbers up to n using recursion :

**Time complexity :** O(n)

**Auxiliary space :** O(n)

**Q4. Sum of Array Elements Using Recursion**

Given an array of integers, find sum of array elements using recursion.

**Examples 1:**

Input : A[] = {1, 2, 3}

Output : 6

1 + 2 + 3 = 6

**Examples 2:**

Input : A[] = {15, 12, 13, 10}

Output : 50

**Q5.To Find Reverse Of String Using Recursion**

Write a recursive function to print the reverse of a given string.

**Examples 1:**

Input : String str=”Hello”

Output : olleH

**Examples 2:**

Input : String str=”Recursion”

Output : noisruceR

**Time Complexity:** O(n)

**Auxiliary Space:** O(n)

**Easy**

**1.** **Merge Two Sorted Lists**

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]

**Constraints:**

The number of nodes in both lists is in the range [0, 50].

-100 <= Node.val <= 100

Both list1 and list2 are sorted in non-decreasing order.

**2.** **Remove Linked List Elements**

Given the head of a linked list and an integer val, remove all the nodes of the linked list that has Node.val == val, and return the new head.

**Example 1:**

Input: head = [1,2,6,3,4,5,6], val = 6

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

**Example 2:**

Input: head = [], val = 1

Output: []

**Constraints:**

The number of nodes in the list is in the range [0, 104].

1 <= Node.val <= 50

0 <= val <= 50

**3.** **Reverse Linked List**

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]

**Constraints:**

The number of nodes in the list is the range [0, 5000].

-5000 <= Node.val <= 5000

Follow up: A linked list can be reversed either iteratively or recursively. Could you implement both?

**4.Power Of Three**

Given an integer n, return true if it is a power of three. Otherwise, return false.

An integer n is a power of three, if there exists an integer x such that n == 3x.

**Example 1:**

Input: n = 27

Output: true

Explanation: 27 = 33

**Example 2:**

Input: n = 0

Output: false

Explanation: There is no x where 3x = 0.

**Constraints:**

-231 <= n <= 231 - 1

Follow up: Could you solve it without loops/recursion?

**5.Palindrome Linked List**

Given the head of a singly linked list, return true if it is a

palindrome

or false otherwise.

**Example 1:**

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

Output: true

**Example 2:**

Input: head = [1,2]

Output: false

**Constraints:**

The number of nodes in the list is in the range [1, 105].

0 <= Node.val <= 9

Follow up: Could you do it in O(n) time and O(1) space?

**6. Find the K-th Character in String Game**

Alice and Bob are playing a game. Initially, Alice has a string word = "a".

You are given a positive integer k

Now Bob will ask Alice to perform the following operation forever:

Generate a new string by changing each character in word to its next character in the English alphabet, and append it to the original word.

For example, performing the operation on "c" generates "cd" and performing the operation on "zb" generates "zbac".

Return the value of the kth character in word, after enough operations have been done for word to have at least k characters.

Note that the character 'z' can be changed to 'a' in the operation.

**Example 1**

Input: k = 5

Output: "b"

Explanation:

Initially, word = "a". We need to do the operation three times:

Generated string is "b", word becomes "ab".

Generated string is "bc", word becomes "abbc".

Generated string is "bccd", word becomes "abbcbccd".

**Example 2:**

Input: k = 10

Output: "c"

**Constraints:**

1 <= k <= 500

**Medium**

**Q1 . Add Two Numbers**

You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

**Example 1:**

Input: l1 = [2,4,3], l2 = [5,6,4]

Output: [7,0,8]

Explanation: 342 + 465 = 807.

**Example 2:**

Input: l1 = [0], l2 = [0]

Output: [0]

**Example 3:**

Input: l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]

Output: [8,9,9,9,0,0,0,1]

**Constraints:**

The number of nodes in each linked list is in the range [1, 100].

0 <= Node.val <= 9

It is guaranteed that the list represents a number that does not have leading zeros.

**Q2 . Elimination Game**

You have a list arr of all integers in the range [1, n] sorted in a strictly increasing order. Apply the following algorithm on arr:

Starting from left to right, remove the first number and every other number afterward until you reach the end of the list.

Repeat the previous step again, but this time from right to left, remove the rightmost number and every other number from the remaining numbers.

Keep repeating the steps again, alternating left to right and right to left, until a single number remains.

Given the integer n, return the last number that remains in arr.

**Example 1:**

Input: n = 9

Output: 6

Explanation:

arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]

arr = [2, 4, 6, 8]

arr = [2, 6]

arr = [6]

**Example 2:**

Input: n = 1

Output: 1

**Constraints:**

1 <= n <= 109

**Q3 . Predict The Winner**

You are given an integer array nums. Two players are playing a game with this array: player 1 and player 2.

Player 1 and player 2 take turns, with player 1 starting first. Both players start the game with a score of 0. At each turn, the player takes one of the numbers from either end of the array (i.e., nums[0] or nums[nums.length - 1]) which reduces the size of the array by 1. The player adds the chosen number to their score. The game ends when there are no more elements in the array.

Return true if Player 1 can win the game. If the scores of both players are equal, then player 1 is still the winner, and you should also return true. You may assume that both players are playing optimally.

**Example 1:**

Input: nums = [1,5,2]

Output: false

Explanation: Initially, player 1 can choose between 1 and 2.

If he chooses 2 (or 1), then player 2 can choose from 1 (or 2) and 5. If player 2 chooses 5, then player 1 will be left with 1 (or 2).

So, final score of player 1 is 1 + 2 = 3, and player 2 is 5.

Hence, player 1 will never be the winner and you need to return false.

**Example 2:**

Input: nums = [1,5,233,7]

Output: true

Explanation: Player 1 first chooses 1. Then player 2 has to choose between 5 and 7. No matter which number player 2 choose, player 1 can choose 233.

Finally, player 1 has more score (234) than player 2 (12), so you need to return True representing player1 can win.

**Constraints:**

1 <= nums.length <= 20

0 <= nums[i] <= 107

**Q4 . Find The Winner Of Circular Game**

There are n friends that are playing a game. The friends are sitting in a circle and are numbered from 1 to n in clockwise order. More formally, moving clockwise from the ith friend brings you to the (i+1)th friend for 1 <= i < n, and moving clockwise from the nth friend brings you to the 1st friend.

The rules of the game are as follows:

Start at the 1st friend.

Count the next k friends in the clockwise direction including the friend you started at. The counting wraps around the circle and may count some friends more than once.

The last friend you counted leaves the circle and loses the game.

If there is still more than one friend in the circle, go back to step 2 starting from the friend immediately clockwise of the friend who just lost and repeat.

Else, the last friend in the circle wins the game.

Given the number of friends, n, and an integer k, return the winner of the game.

**Example 1:**

Input: n = 5, k = 2

Output: 3

Explanation: Here are the steps of the game:

1) Start at friend 1.

2) Count 2 friends clockwise, which are friends 1 and 2.

3) Friend 2 leaves the circle. Next start is friend 3.

4) Count 2 friends clockwise, which are friends 3 and 4.

5) Friend 4 leaves the circle. Next start is friend 5.

6) Count 2 friends clockwise, which are friends 5 and 1.

7) Friend 1 leaves the circle. Next start is friend 3.

8) Count 2 friends clockwise, which are friends 3 and 5.

9) Friend 5 leaves the circle. Only friend 3 is left, so they are the winner.

**Example 2:**

Input: n = 6, k = 5

Output: 1

Explanation: The friends leave in this order: 5, 4, 6, 2, 3. The winner is friend 1.

**Constraints:**

1 <= k <= n <= 500

Follow up:

Could you solve this problem in linear time with constant space?

**Q5 Minimun non zero product of an Array Elements**

You are given a positive integer p. Consider an array nums (1-indexed) that consists of the integers in the inclusive range [1, 2p - 1] in their binary representations. You are allowed to do the following operation any number of times:

Choose two elements x and y from nums.

Choose a bit in x and swap it with its corresponding bit in y. Corresponding bit refers to the bit that is in the same position in the other integer.

For example, if x = 1101 and y = 0011, after swapping the 2nd bit from the right, we have x = 1111 and y = 0001.

Find the minimum non-zero product of nums after performing the above operation any number of times. Return this product modulo 109 + 7.

Note: The answer should be the minimum product before the modulo operation is done.

**Example 1:**

Input: p = 1

Output: 1

Explanation: nums = [1].

There is only one element, so the product equals that element.

**Example 2:**

Input: p = 2

Output: 6

Explanation: nums = [01, 10, 11].

Any swap would either make the product 0 or stay the same.

Thus, the array product of 1 \* 2 \* 3 = 6 is already minimized.

**Example 3:**

Input: p = 3

Output: 1512

Explanation: nums = [001, 010, 011, 100, 101, 110, 111]

- In the first operation we can swap the leftmost bit of the second and fifth elements.

- The resulting array is [001, 110, 011, 100, 001, 110, 111].

- In the second operation we can swap the middle bit of the third and fourth elements.

- The resulting array is [001, 110, 001, 110, 001, 110, 111].

The array product is 1 \* 6 \* 1 \* 6 \* 1 \* 6 \* 7 = 1512, which is the minimum possible product.

**Constraints:**

1 <= p <= 60

**Hard**

**Q1 ..Regular Expression Matching**

Given an input string s and a pattern p, implement regular expression matching with support for '.' and '\*' where:

'.' Matches any single character.​​​​

'\*' Matches zero or more of the preceding element.

The matching should cover the entire input string (not partial).

**Example 1:**

Input: s = "aa", p = "a"

Output: false

Explanation: "a" does not match the entire string "aa".

**Example 2:**

Input: s = "aa", p = "a\*"

Output: true

Explanation: '\*' means zero or more of the preceding element, 'a'. Therefore, by repeating 'a' once, it becomes "aa".

**Example 3:**

Input: s = "ab", p = ".\*"

Output: true

Explanation: ".\*" means "zero or more (\*) of any character (.)".

**Constraints:**

1 <= s.length <= 20

1 <= p.length <= 20

s contains only lowercase English letters.

p contains only lowercase English letters, '.', and '\*'.

It is guaranteed for each appearance of the character '\*', there will be a previous valid character to match.

**Q2. Reverse Nodes in k-Group**

Given the head of a linked list, reverse the nodes of the list k at a time, and return the modified list.

k is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of k then left-out nodes, in the end, should remain as it is.

You may not alter the values in the list's nodes, only nodes themselves may be changed.

**Example 1:**

Input: head = [1,2,3,4,5], k = 2

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

**Example 2:**

Input: head = [1,2,3,4,5], k = 3

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

**Constraints:**

The number of nodes in the list is n.

1 <= k <= n <= 5000

0 <= Node.val <= 1000

Follow-up: Can you solve the problem in O(1) extra memory space?

**Q3. Wildcard Matching**

Given an input string (s) and a pattern (p), implement wildcard pattern matching with support for '?' and '\*' where:

'?' Matches any single character.

'\*' Matches any sequence of characters (including the empty sequence).

The matching should cover the entire input string (not partial).

**Example 1:**

Input: s = "aa", p = "a"

Output: false

Explanation: "a" does not match the entire string "aa".

**Example 2:**

Input: s = "aa", p = "\*"

Output: true

Explanation: '\*' matches any sequence.

**Example 3:**

Input: s = "cb", p = "?a"

Output: false

Explanation: '?' matches 'c', but the second letter is 'a', which does not match 'b'.

**Constraints:**

0 <= s.length, p.length <= 2000

s contains only lowercase English letters.

p contains only lowercase English letters, '?' or '\*'.

**Q4. Permutation Sequence**

The set [1, 2, 3, ..., n] contains a total of n! unique permutations.

By listing and labeling all of the permutations in order, we get the following sequence for n = 3:

"123"

"132"

"213"

"231"

"312"

"321"

Given n and k, return the kth permutation sequence.

**Example 1:**

Input: n = 3, k = 3

Output: "213"

**Example 2:**

Input: n = 4, k = 9

Output: "2314"

**Example 3:**

Input: n = 3, k = 1

Output: "123"

**Constraints:**

1 <= n <= 9

1 <= k <= n!

**Q5. Basic Calculator**

Given a string s representing a valid expression, implement a basic calculator to evaluate it, and return the result of the evaluation.

Note: You are not allowed to use any built-in function which evaluates strings as mathematical expressions, such as eval().

**Example 1**:

Input: s = "1 + 1"

Output: 2

**Example 2:**

Input: s = " 2-1 + 2 "

Output: 3

**Example 3:**

Input: s = "(1+(4+5+2)-3)+(6+8)"

Output: 23

**Constraints:**

1 <= s.length <= 3 \* 105

s consists of digits, '+', '-', '(', ')', and ' '.

s represents a valid expression.

'+' is not used as a unary operation (i.e., "+1" and "+(2 + 3)" is invalid).

'-' could be used as a unary operation (i.e., "-1" and "-(2 + 3)" is valid).

There will be no two consecutive operators in the input.

Every number and running calculation will fit in a signed 32-bit integer.

**Very Hard**

**Q1. Find the K-th Character in String Game II**

Alice and Bob are playing a game. Initially, Alice has a string word = "a".

You are given a positive integer k. You are also given an integer array operations, where operations[i] represents the type of the ith operation.

Now Bob will ask Alice to perform all operations in sequence:

If operations[i] == 0, append a copy of word to itself.

If operations[i] == 1, generate a new string by changing each character in word to its next character in the English alphabet, and append it to the original word. For example, performing the operation on "c" generates "cd" and performing the operation on "zb" generates "zbac".

Return the value of the kth character in word after performing all the operations.

Note that the character 'z' can be changed to 'a' in the second type of operation.

**Example 1:**

Input: k = 5, operations = [0,0,0]

Output: "a"

Explanation:

Initially, word == "a". Alice performs the three operations as follows:

Appends "a" to "a", word becomes "aa".

Appends "aa" to "aa", word becomes "aaaa".

Appends "aaaa" to "aaaa", word becomes "aaaaaaaa".

**Example 2:**

Input: k = 10, operations = [0,1,0,1]

Output: "b"

Explanation:

Initially, word == "a". Alice performs the four operations as follows:

Appends "a" to "a", word becomes "aa".

Appends "bb" to "aa", word becomes "aabb".

Appends "aabb" to "aabb", word becomes "aabbaabb".

Appends "bbccbbcc" to "aabbaabb", word becomes "aabbaabbbbccbbcc".

**Constraints:**

1 <= k <= 1014

1 <= operations.length <= 100

operations[i] is either 0 or 1.

The input is generated such that word has at least k characters after all operations.

**Q2. Maximize Number of Nice Divisors**

You are given a positive integer primeFactors. You are asked to construct a positive integer n that satisfies the following conditions:

The number of prime factors of n (not necessarily distinct) is at most primeFactors.

The number of nice divisors of n is maximized. Note that a divisor of n is nice if it is divisible by every prime factor of n. For example, if n = 12, then its prime factors are [2,2,3], then 6 and 12 are nice divisors, while 3 and 4 are not.

Return the number of nice divisors of n. Since that number can be too large, return it modulo 109 + 7.

Note that a prime number is a natural number greater than 1 that is not a product of two smaller natural numbers. The prime factors of a number n is a list of prime numbers such that their product equals n.

**Example 1:**

Input: primeFactors = 5

Output: 6

Explanation: 200 is a valid value of n.

It has 5 prime factors: [2,2,2,5,5], and it has 6 nice divisors: [10,20,40,50,100,200].

There is not other value of n that has at most 5 prime factors and more nice divisors.

**Example 2:**

Input: primeFactors = 8

Output: 18

**Constraints:**

1 <= primeFactors <= 109

**Q3. Parsing a Boolean Expression**

A boolean expression is an expression that evaluates to either true or false. It can be in one of the following shapes:

't' that evaluates to true.

'f' that evaluates to false.

'!(subExpr)' that evaluates to the logical NOT of the inner expression subExpr.

'&(subExpr1, subExpr2, ..., subExprn)' that evaluates to the logical AND of the inner expressions subExpr1, subExpr2, ..., subExprn where n >= 1.

'|(subExpr1, subExpr2, ..., subExprn)' that evaluates to the logical OR of the inner expressions subExpr1, subExpr2, ..., subExprn where n >= 1.

Given a string expression that represents a boolean expression, return the evaluation of that expression.

It is guaranteed that the given expression is valid and follows the given rules.

**Example 1:**

Input: expression = "&(|(f))"

Output: false

Explanation:

First, evaluate |(f) --> f. The expression is now "&(f)".

Then, evaluate &(f) --> f. The expression is now "f".

Finally, return false.

**Example 2:**

Input: expression = "|(f,f,f,t)"

Output: true

Explanation: The evaluation of (false OR false OR false OR true) is true.

**Example 3:**

Input: expression = "!(&(f,t))"

Output: true

Explanation:

First, evaluate &(f,t) --> (false AND true) --> false --> f. The expression is now "!(f)".

Then, evaluate !(f) --> NOT false --> true. We return true.

**Constraints:**

1 <= expression.length <= 2 \* 104

expression[i] is one following characters: '(', ')', '&', '|', '!', 't', 'f', and ','.

**Q4 . Special Binary String**

Special binary strings are binary strings with the following two properties:

The number of 0's is equal to the number of 1's.

Every prefix of the binary string has at least as many 1's as 0's.

You are given a special binary string s.

A move consists of choosing two consecutive, non-empty, special substrings of s, and swapping them. Two strings are consecutive if the last character of the first string is exactly one index before the first character of the second string.

Return the lexicographically largest resulting string possible after applying the mentioned operations on the string.

**Example 1:**

Input: s = "11011000"

Output: "11100100"

Explanation: The strings "10" [occuring at s[1]] and "1100" [at s[3]] are swapped.

This is the lexicographically largest string possible after some number of swaps.

**Example 2:**

Input: s = "10"

Output: "10"

**Constraints:**

1 <= s.length <= 50

s[i] is either '0' or '1'.

s is a special binary string.

**Q5 . Parse Lisp Expression**

You are given a string expression representing a Lisp-like expression to return the integer value of.

The syntax for these expressions is given as follows.

An expression is either an integer, let expression, add expression, mult expression, or an assigned variable. Expressions always evaluate to a single integer.

(An integer could be positive or negative.)

A let expression takes the form "(let v1 e1 v2 e2 ... vn en expr)", where let is always the string "let", then there are one or more pairs of alternating variables and expressions, meaning that the first variable v1 is assigned the value of the expression e1, the second variable v2 is assigned the value of the expression e2, and so on sequentially; and then the value of this let expression is the value of the expression expr.

An add expression takes the form "(add e1 e2)" where add is always the string "add", there are always two expressions e1, e2 and the result is the addition of the evaluation of e1 and the evaluation of e2.

A mult expression takes the form "(mult e1 e2)" where mult is always the string "mult", there are always two expressions e1, e2 and the result is the multiplication of the evaluation of e1 and the evaluation of e2.

For this question, we will use a smaller subset of variable names. A variable starts with a lowercase letter, then zero or more lowercase letters or digits. Additionally, for your convenience, the names "add", "let", and "mult" are protected and will never be used as variable names.

Finally, there is the concept of scope. When an expression of a variable name is evaluated, within the context of that evaluation, the innermost scope (in terms of parentheses) is checked first for the value of that variable, and then outer scopes are checked sequentially. It is guaranteed that every expression is legal. Please see the examples for more details on the scope.

**Example 1:**

Input: expression = "(let x 2 (mult x (let x 3 y 4 (add x y))))"

Output: 14

Explanation: In the expression (add x y), when checking for the value of the variable x,

we check from the innermost scope to the outermost in the context of the variable we are trying to evaluate.

Since x = 3 is found first, the value of x is 3.

**Example 2:**

Input: expression = "(let x 3 x 2 x)"

Output: 2

Explanation: Assignment in let statements is processed sequentially.

**Example 3:**

Input: expression = "(let x 1 y 2 x (add x y) (add x y))"

Output: 5

Explanation: The first (add x y) evaluates as 3, and is assigned to x.

The second (add x y) evaluates as 3+2 = 5.

**Constraints:**

1 <= expression.length <= 2000

There are no leading or trailing spaces in expression.

All tokens are separated by a single space in expression.

The answer and all intermediate calculations of that answer are guaranteed to fit in a 32-bit integer.

The expression is guaranteed to be legal and evaluate to an integer.

import java.util.\*;

Function Questions

Very Easy

**Q1..Reverse An Array Using Function**

An array is a type of data structure that stores elements of the same type in a contiguous block of memory. In an array A , of size N, each memory location has some unique index i , (where 0<=i<=N), that can be referenced as or A[i] or Ai

Reverse an array of integers. (Using Function)

**Example:**A=[1,2,3]

Return . [3,2,1]

Function Description

Complete the function reverseArray in the editor below.

reverseArray has the following parameter(s):

* int A[n]: the array to reverse

Returns

* int[n]: the reversed array

Input Format

The first line contains an integer N, , the number of integers in A.  
The second line contains N space-separated integers that make up A.

**Constraints:**

1<=N<=10^3

1<=A[i]<=10^4

**Q2. Write a Function to print first name and last name using function**

You are given the firstname and lastname of a person on two different lines. Your task is to read them and print the following using function:

Hello firstname lastname! You just delved into function.

**Function Description:**

Complete the print\_full\_name function in the editor below.

print\_full\_name has the following parameters:

string first: the first name

string last: the last name

**Prints**

string: 'Hello firstname lastname ! You just delved into using the function' where firstname and lastname are replaced with first and last.

**Input Format**

The first line contains the first name, and the second line contains the last name.

**Constraints**

The length of the first and last names are each ≤10 .

**Sample Input 0**

Ross

Taylor

**Sample Output 0**

Hello Ross Taylor! You just delved into function

**Explanation 0**

The input read by the program is stored as a string data type. A string is a collection of characters.

Q3 . Maximum of Two Numbers Using Function

Imagine you are developing a calculator app where users can compare two numbers to determine the larger one. For example, in a competitive game, players want to know who scored the higher points. You need to implement a function that takes two integers, compares them, and returns the greater value. This function can then be reused across various parts of the app for comparison tasks.

**Description:**Given two integers, write a function that returns the larger of the two numbers.

**Example 1:**

Input: num1 = 10, num2 = 20

Output: 20

Explanation: 20 is larger than 10.

**Example 2:**

Input: num1 = -5, num2 = -10

Output: -5

Explanation: -5 is larger than -10.

**Q4. Calculate the Area of a Rectangle Using Function**

Imagine you are developing a real-estate management app that calculates the floor area of different rooms. A user provides the length and width of a rectangle-shaped room, and your app needs to calculate its area. This function can also be applied in construction projects, interior designing, or other applications where accurate space measurement is essential. Your task is to write a reusable function for this calculation.

**Description**:

Given the length and width of a rectangle, write a function to calculate its area. The area of a rectangle is defined as :

Area=Length×Width.

**Example 1:**

Input: length = 5, width = 10

Output: 50

Explanation: The area is calculated as 5 × 10 = 50.

**Example 2:**

Input: length = 7, width = 3

Output: 21

Explanation: The area is calculated as 7 × 3 = 21.

**Q5 Create Function to Check Given Number is Armstrong or not**

GK given you a number *n*.Your task is to find whether the number is Armstrong or not.

**INPUT:**

The input consists of only single line containing number *n.*

**OUTPUT:**

Print 1 if the number is Armstrong .Otherwise print 0.

**CONSTRAINT:**

1<n<105

**Easy**

**Q1 Find GCD of Number Using Function**

Given an integer array nums, return the greatest common divisor of the smallest number and largest number in nums.

The greatest common divisor of two numbers is the largest positive integer that evenly divides both numbers.

**Example 1:**

Input: nums = [2,5,6,9,10]

Output: 2

Explanation:

The smallest number in nums is 2.

The largest number in nums is 10.

The greatest common divisor of 2 and 10 is 2.

**Example 2:**

Input: nums = [7,5,6,8,3]

Output: 1

Explanation:

The smallest number in nums is 3.

The largest number in nums is 8.

The greatest common divisor of 3 and 8 is 1.

**Example 3:**

Input: nums = [3,3]

Output: 3

Explanation:

The smallest number in nums is 3.

The largest number in nums is 3.

The greatest common divisor of 3 and 3 is 3.

**Constraints:**

2 <= nums.length <= 1000

1 <= nums[i] <= 1000

**Q2. Sort an Array using function (do not use any built in function )**

Given an array of integers nums, sort the array in ascending order and return it.

You must solve the problem without using any built-in functions in O(nlog(n)) time complexity and with the smallest space complexity possible.

**Example 1:**

Input: nums = [5,2,3,1]

Output: [1,2,3,5]

Explanation: After sorting the array, the positions of some numbers are not changed (for example, 2 and 3), while the positions of other numbers are changed (for example, 1 and 5).

**Example 2:**

Input: nums = [5,1,1,2,0,0]

Output: [0,0,1,1,2,5]

Explanation: Note that the values of nums are not necessairly unique.

**Constraints:**

1 <= nums.length <= 5 \* 104

-5 \* 104 <= nums[i] <= 5 \* 104

**Q3 . Peek at the Top Element**

Imagine you are designing a text editor with an "undo" feature. Each action the user performs is stored in a stack. Before the user decides to undo the last action, the editor needs to display the most recent action without actually removing it from the stack. This ensures that the user can preview their last action before committing to undo. To implement this functionality, you need a function that retrieves the top element of the stack without modifying it.

Write a function to return the top element of a stack without removing it.

Input: Stack.

Output: The top element

**Q4. Pre-order Traversal of a Binary Tree Using Function**

You are designing a file system explorer, where files and folders are represented as nodes in a tree structure. In this tree, folders can contain other folders or files, and files are leaf nodes. To display the contents of the file system starting from the root folder, you need to visit each folder before its contents in a depth-first manner. This requires implementing a **pre-order traversal** of the binary tree structure representing the file system. This traversal ensures that each parent folder is processed before its children are explored.

**Input**: root = [1,null,2,3]

**Output**: [1,2,3]

The pre-order traversal visits nodes in the order: 1 → 2 → 3.

**Constraints:**

The number of nodes in the tree is in the range [0,10^4]

-10^4<=Node.val<=10^4

Follow-up:

Can you solve this problem iteratively instead of using recursion?

**Q5.. Enqueue an Element**

You are building a ticketing system for a movie theater where customers line up in a queue to buy tickets. As new customers arrive, they are added to the end of the queue. When it's a customer's turn to buy a ticket, they are removed from the front of the queue. You need to implement a function that allows new customers to join the queue by adding them to the end, ensuring the system maintains the correct order.

**Description:**

Write a function to add an element to the end of a queue. The function should modify the queue by appending the given element at the rear of the queue.

Input: queue = [1, 2, 3], element = 4

Output: [1, 2, 3, 4]

Explanation: The element 4 is added to the end of the queue.

**Constraints:**

* The number of elements in the queue is between 0≤length of queue≤1040 \leq \text{length of queue} \leq 10^40≤length of queue≤104.
* The element to be added can be any integer.

**Follow-up:**Can you implement this function using a dynamic array or a linked list for better performance with large queues?

**Medium**

### **Q1 : Add an Edge to an Undirected Graph**

**Description:**Write a function to add an edge between two nodes in an undirected graph. The graph is represented as an adjacency list. When two nodes are connected by an edge, both nodes should reference each other.

**Input:**

* A graph represented as an adjacency list (dictionary or list of lists).
* Two nodes node1 and node2 representing the users being connected.

**Output:**

* The updated graph with the added edge between node1 and node2.

**Example 1:**

Input:

graph = {1: [2], 2: [1]}

node1 = 2, node2 = 3

Output:

graph = {1: [2], 2: [1, 3], 3: [2]}

Explanation: The edge between 2 and 3 is added.

**Constraints:**

* The graph is undirected, meaning if there's an edge from node1 to node2, there should also be an edge from node2 to node1.
* Nodes are represented by integers, and the graph is initially empty or partially filled with connections.

**Follow-up:**How would you implement this function using an adjacency matrix instead of an adjacency list?

**Q2.: Longest Substring Without Repeating Characters**

You are working on building a text editor application. One of the features you're focusing on is a "word suggestion" system. The editor needs to identify the longest sequence of characters typed without repeating any letters to suggest potential words or phrases. To accomplish this, you must efficiently find the length of the longest substring of unique characters as the user types.

**Description:**Write a function that takes a string as input and returns the length of the longest substring without repeating characters. A substring is a contiguous sequence of characters within the string.

**Example 1:**

Input: "abcabcbb"

Output: 3

Explanation: The longest substring without repeating characters is "abc", which has length 3.

**Constraints:**

* The input string will have a length between 111 and 10410^4104.
* The characters in the string are printable ASCII characters.

This problem is a classic example of the **sliding window** technique,

**Q3: Evaluate Reverse Polish Notation**

You are building a calculator application that evaluates mathematical expressions. The calculator uses Reverse Polish Notation (RPN), also known as Postfix notation, where operators follow their operands. This eliminates the need for parentheses and operator precedence rules. For example, the expression 2 3 + is equivalent to 2 + 3 in infix notation. Your task is to write a function that evaluates an expression in RPN format.

**Description:**Write a function to evaluate the value of an expression given in Reverse Polish Notation. The expression is a list of strings representing operands and operators. You need to compute the result by applying the operators in the order they appear.

**Input:**

* A list of tokens (strings) that represent the expression in Reverse Polish Notation.  
  Each token is either an operand (integer) or an operator (+, -, \*, /).

**Output:**

* The result of evaluating the expression.

**Example 1:**

Input: ["2", "1", "+", "3", "\*"]

Output: 9

Explanation: (2 + 1) \* 3 = 9

**Constraints:**

* The input list will have between 1 and 1000 tokens.
* Tokens will be integers or operators from the set {"+", "-", "\*", "/"}.
* The division operator / should truncate toward zero.

**Q4. Flatten a Binary Tree to Linked List**

You are working on a tree data structure library and are tasked with implementing a function to flatten a binary tree into a linked list. The flattening should be done **in-place**, meaning you cannot use extra memory for the linked list structure but must modify the tree itself. The tree should be flattened such that it appears as a **right-skewed linked list** (i.e., all left child nodes should be null, and each node's right child should point to the next node in the pre-order traversal).

**Description:**Write a function that flattens a binary tree into a "linked list" in-place. After flattening, the left child of each node should be set to null, and the right child should point to the next node in the pre-order traversal of the tree.

**Input:**

* The root node of a binary tree.

**Output:**

* The flattened binary tree.

**Example 1:**

Input:

1

/ \

2 5

/ \ \

3 4 6

Output:

1 -> 2 -> 3 -> 4 -> 5 -> 6

**Explanation:**The tree is flattened in a way that the left child pointers are all null, and the right child pointers point to the next node in pre-order traversal.

**Constraints:**

* The tree will have at most 10410^4104 nodes.
* Each node will have an integer value.

**Q 5: Flatten a Binary Tree to Linked List**

You are working on a tree manipulation module for a data processing system. The system requires hierarchical data structures (binary trees) to be transformed into linear sequences for easier processing. The task is to flatten a given binary tree in such a way that all nodes follow the order of a **pre-order traversal**, with no left children and all nodes connected through their right child pointers. The transformation should be done **in-place** without using additional data structures.

**Description:**Given the root of a binary tree, flatten it to a "linked list" in-place. After flattening, each node's left child should point to null, and the right child should point to the next node in pre-order traversal.

**Input:**

A TreeNode object representing the root of the binary tree. Each TreeNode has the following structure:  
python  
Copy code  
class TreeNode:

def \_\_init\_\_(self, val=0, left=None, right=None):

self.val = val

self.left = left

self.right = right

**Output:**

* The root of the same tree, now modified as a flattened linked list.

### **Examples**

**Example 1:**Input:

markdown

Copy code

1

/ \

2 5

/ \ \

3 4 6

Output:

1 -> 2 -> 3 -> 4 -> 5 -> 6

Explanation:  
After flattening, the tree becomes a single right-skewed structure.

### **Constraints:**

1. The number of nodes in the tree is between 111 and 10410^4104.
2. The tree is a binary tree (each node has at most two children).
3. The values of the nodes are integers.

### **Follow-Up:**

* Can you achieve the solution with O(1)O(1)O(1) additional space while maintaining O(n)O(n)O(n) time complexity? (Hint: Think about Morris Traversal or modifying the tree structure in-place during recursion.)

**Hard**

**Q1: Maximum Subarray Product**

You are developing a financial analysis application where users input daily stock price changes in an array. Your task is to determine the maximum profit or loss achievable over a contiguous period, considering that the profit/loss for a period is calculated by multiplying the daily percentage changes. This problem is critical for finding optimal periods to make decisions for investments or short-term trading.

**Description:**Write a function that takes an integer array as input and returns the maximum product of a contiguous subarray. The array can contain both positive and negative integers, and your function must account for these to find the optimal subarray.

**Input:**

* An array of integers nums[]nums[]nums[] where 1≤nums.length≤1041 \leq nums.length \leq 10^41≤nums.length≤104 and −10≤nums[i]≤10-10 \leq nums[i] \leq 10−10≤nums[i]≤10.

**Output:**

* An integer representing the maximum product of any contiguous subarray.

**Example 1:**

Input: nums = [2, 3, -2, 4]

Output: 6

Explanation: The subarray [2, 3] has the largest product = 6.

### **Constraints:**

1. The input array may contain both positive and negative numbers, including zero.
2. You must solve the problem with a time complexity of O(n)O(n)O(n).

**Q2.Minimum Window Substring**

You are building a text analysis feature for a document editor. A user selects a set of keywords (represented as string t) and wants to find the shortest snippet (substring) in their document (string s) that contains all those keywords, regardless of order. This feature is crucial for tasks such as highlighting relevant sections in large documents or extracting concise summaries.

**Description:**Write a function to find the smallest substring in string s that contains all the characters of another string t. If no such substring exists, return an empty string.

**Input:**

* A string s of length 1≤∣s∣≤1051 \leq |s| \leq 10^51≤∣s∣≤105, consisting of lowercase letters.
* A string t of length 1≤∣t∣≤∣s∣1 \leq |t| \leq |s|1≤∣t∣≤∣s∣, consisting of lowercase letters.

**Output:**

* The smallest substring of s containing all the characters of t. If there are multiple results, return the one that appears first.

**Example 1:**

Input: s = "ADOBECODEBANC", t = "ABC"

Output: "BANC"

Explanation: The substring "BANC" is the shortest substring of `s` that contains all characters of `t`

### **Constraints:**

1. If t contains duplicate characters, those characters must appear in the substring with at least the same frequency.
2. The order of characters in t does not matter for the substring.
3. If no substring satisfies the condition, return an empty string.

Q3 **Linked List: Copy List with Random Pointers**

**Scenario:**You are implementing a simulation system where entities have connections to other entities represented by random pointers. You need to create an independent copy of these entities while preserving their original structure and relationships.

**Problem Statement:**Write a function to create a deep copy of a linked list where each node contains:

1. A value (val),
2. A pointer to the next node (next),
3. A random pointer (random) that can point to any node in the list or be null.

**Constraints:**

* The linked list may be empty or contain cycles via the random pointer.  
  **Input:**
* The head of a linked list where each node has val, next, and random properties.  
  **Output:**
* The head of the deep-copied linked list.

**Examples:**

Input: Linked list with nodes [7 -> 13 -> 11 -> 10 -> 1], where:

Node 13.random points to 7,

Node 11.random points to 1,

Node 10.random points to 11.

Output: A deep copy of the original linked list with identical structure.

**Q4.Redesigned Question: Serialize and Deserialize Binary Tree**

You are designing a data storage system that needs to save and retrieve hierarchical data, such as file structures or family trees. The system must encode a binary tree into a format that can be stored as a string and then decode it back into the original tree structure. This is critical for efficient saving, sharing, and reconstruction of complex tree-like data.

Implement two functions:

1. serialize(root): Encodes a binary tree to a single string.
2. deserialize(data): Decodes the encoded string back into the original binary tree.

**Constraints:**

* The binary tree can contain between 111 and 10410^4104 nodes.
* Node values are integers in the range [−104,104][-10^4, 10^4][−104,104].

**Input:**

* For serialize(): The root node of a binary tree.
* For deserialize(): A string representing the serialized tree.

**Output:**

* For serialize(): A string that represents the binary tree.
* For deserialize(): The reconstructed binary tree.

### 

**Example 1:**

Input (Tree): [1,2,3,null,null,4,5]

Serialized Output: "1,2,3,null,null,4,5"

Deserialized Output (Tree): [1,2,3,null,null,4,5]

### **Q5: Word Ladder II**

**Scenario:**You are developing a word puzzle game where players transform one word into another by changing one letter at a time. The goal is to find all possible shortest transformation paths. Each intermediate word must be a valid dictionary word, and the transformation process must terminate at the target word. This problem is central to applications like word games and linguistic analysis tools.

**Description:**Write a function to find all the shortest transformation sequences from a given beginWord to endWord, such that:

1. Only one letter can be changed at a time.
2. All intermediate words must exist in a provided dictionary (wordList).

**Constraints:**

* The length of all words (including beginWord and endWord) is the same: 1≤word length≤101 \leq \text{word length} \leq 101≤word length≤10.
* The dictionary size is up to 10410^4104.
* Words in the dictionary are unique, and the dictionary may not contain duplicates of beginWord or endWord.
* If no transformation sequence is possible, return an empty list.

**Input:**

* A string beginWord.
* A string endWord.
* A list of strings wordList.

**Output:**

* A list of all shortest transformation sequences (each sequence is a list of words).

**Example 1:**

Input:

beginWord = "hit",

endWord = "cog",

wordList = ["hot","dot","dog","lot","log","cog"]

Output:

[

["hit","hot","dot","dog","cog"],

["hit","hot","lot","log","cog"]

]

Explanation:

There are two shortest transformation sequences from "hit" to "cog", each with a length of 5.

import java.util.\*;

**Very Hard**

Q1 There are n children standing in a line. Each child is assigned a rating value given in the integer array ratings.

You are giving candies to these children subjected to the following requirements:

Each child must have at least one candy.

Children with a higher rating get more candies than their neighbors.

Return the minimum number of candies you need to have to distribute the candies to the children.

**Example 1:**

Input: ratings = [1,0,2]

Output: 5

Explanation: You can allocate to the first, second and third child with 2, 1, 2 candies respectively.

**Example 2:**

Input: ratings = [1,2,2]

Output: 4

Explanation: You can allocate to the first, second and third child with 1, 2, 1 candies respectively.

The third child gets 1 candy because it satisfies the above two conditions.

**Constraints:**

n == ratings.length

1 <= n <= 2 \* 104

0 <= ratings[i] <= 2 \* 104

**Q2 Word Break**

Given a string s and a dictionary of strings wordDict, add spaces in s to construct a sentence where each word is a valid dictionary word. Return all such possible sentences in any order.

Note that the same word in the dictionary may be reused multiple times in the segmentation.

**Example 1:**

Input: s = "catsanddog", wordDict = ["cat","cats","and","sand","dog"]

Output: ["cats and dog","cat sand dog"]

**Example 2:**

Input: s = "pineapplepenapple", wordDict = ["apple","pen","applepen","pine","pineapple"]

Output: ["pine apple pen apple","pineapple pen apple","pine applepen apple"]

Explanation: Note that you are allowed to reuse a dictionary word.

**Example 3:**

Input: s = "catsandog", wordDict = ["cats","dog","sand","and","cat"]

Output: []

**Constraints:**

1 <= s.length <= 20

1 <= wordDict.length <= 1000

1 <= wordDict[i].length <= 10

s and wordDict[i] consist of only lowercase English letters.

All the strings of wordDict are unique.

Input is generated in a way that the length of the answer doesn't exceed 105.

**Q3** [**Flatten a Multilevel Doubly Linked Lis**](https://leetcode.com/problems/flatten-a-multilevel-doubly-linked-list/)**t**

You are given a doubly linked list, which contains nodes that have a next pointer, a previous pointer, and an additional child pointer. This child pointer may or may not point to a separate doubly linked list, also containing these special nodes. These child lists may have one or more children of their own, and so on, to produce a multilevel data structure as shown in the example below.

Given the head of the first level of the list, flatten the list so that all the nodes appear in a single-level, doubly linked list. Let curr be a node with a child list. The nodes in the child list should appear after curr and before curr.next in the flattened list.

Return the head of the flattened list. The nodes in the list must have all of their child pointers set to null.

**Example 1:**

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

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

Explanation: The multilevel linked list in the input is shown.

After flattening the multilevel linked list it becomes:

**Example 2:**

Input: head = [1,2,null,3]

Output: [1,3,2]

Explanation: The multilevel linked list in the input is shown.

After flattening the multilevel linked list it becomes:

**Example 3:**

Input: head = []

Output: []

Explanation: There could be empty list in the input.

**Constraints:**

The number of Nodes will not exceed 1000.

1 <= Node.val <= 105

**Q4.**[**Cheapest Flights Within K Stop**](https://leetcode.com/problems/cheapest-flights-within-k-stops/)

There are n cities connected by some number of flights. You are given an array flights where flights[i] = [fromi, toi, pricei] indicates that there is a flight from city fromi to city toi with cost pricei.

You are also given three integers src, dst, and k, return the cheapest price from src to dst with at most k stops. If there is no such route, return -1.

**Example 1:**

Input: n = 4, flights = [[0,1,100],[1,2,100],[2,0,100],[1,3,600],[2,3,200]], src = 0, dst = 3, k = 1

Output: 700

Explanation:

The graph is shown above.

The optimal path with at most 1 stop from city 0 to 3 is marked in red and has cost 100 + 600 = 700.

Note that the path through cities [0,1,2,3] is cheaper but is invalid because it uses 2 stops.

**Example 2:**

Input: n = 3, flights = [[0,1,100],[1,2,100],[0,2,500]], src = 0, dst = 2, k = 1

Output: 200

Explanation:

The graph is shown above.

The optimal path with at most 1 stop from city 0 to 2 is marked in red and has cost 100 + 100 = 200.

**Example 3:**

Input: n = 3, flights = [[0,1,100],[1,2,100],[0,2,500]], src = 0, dst = 2, k = 0

Output: 500

Explanation:

The graph is shown above.

The optimal path with no stops from city 0 to 2 is marked in red and has cost 500.

**Constraints:**

1 <= n <= 100

0 <= flights.length <= (n \* (n - 1) / 2)

flights[i].length == 3

0 <= fromi, toi < n

fromi != toi

1 <= pricei <= 104

There will not be any multiple flights between two cities.

0 <= src, dst, k < n

src != dst

**Q5 .Minimum Number of Swaps to Make the String Balanced**

You are given a 0-indexed string s of even length n. The string consists of exactly n / 2 opening brackets '[' and n / 2 closing brackets ']'.

A string is called balanced if and only if:

It is the empty string, or

It can be written as AB, where both A and B are balanced strings, or

It can be written as [C], where C is a balanced string.

You may swap the brackets at any two indices any number of times.

Return the minimum number of swaps to make it balanced.

**Example 1:**

Input: s = "][]["

Output: 1

Explanation: You can make the string balanced by swapping index 0 with index 3.

The resulting string is "[[]]".

**Example 2:**

Input: s = "]]][[["

Output: 2

Explanation: You can do the following to make the string balanced:

- Swap index 0 with index 4. s = "[]][][".

- Swap index 1 with index 5. s = "[[][]]".

The resulting string is "[[][]]".

**Example 3:**

Input: s = "[]"

Output: 0

Explanation: The string is already balanced.

**Constraints:**

n == s.length

2 <= n <= 106

n is even.

s[i] is either '[' or ']'.

The number of opening brackets '[' equals n / 2, and the number of closing brackets ']' equals n / 2.