Exercise No. : **5**

Topics Covered : **Arraylist, Control Flow Statements, Stack, Array, Priority Queue, Linked List, Hashset**

Date : **20-05-2024**

Solve the following problems

Q No.	**Question Detail**	**Level**
1	**Magical Pattern** **Problem statement :** You have been given an integer 'N'. Your task is to print the Magical Pattern(see examples) for the given 'N'. **Example :** For 'N' : 4 Pattern : 4 3 2 1 2 3 4 3 3 2 1 2 3 3 2 2 2 1 2 2 2 1 1 1 1 1 1 1 2 2 2 1 2 2 2 3 3 2 1 2 3 3 4 3 2 1 2 3 4 **Sample Input 1 :** 1 3 **Sample Output 1 :** 3 2 1 2 3 2 2 1 2 2 1 1 1 1 1 2 2 1 2 2 3 2 1 2 3 **Sample Input 2 :** 1 5 **Sample Output 2 :**	Easy

	5 4 3 2 1 2 3 4 5 4 4 3 2 1 2 3 4 4 3 3 3 2 1 2 3 3 3 2 2 2 2 1 2 2 2 2 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 2 2 2 3 3 3 2 1 2 3 3 3 4 4 3 2 1 2 3 4 4 5 4 3 2 1 2 3 4 5 **Constraints :** • 1<= T <= 10 • 1<= N <= 10^2	
2	**Interesting Alphabets** **Problem statement :** As a part of its competition, the school will conduct a codeathon, Lock the Code, where it has been given a value, and the participants have to decode it. The participants are given a value denoting the number of rows in the matrix; they need to print the pattern. **Example :** For N=5, Pattern: E DE CDE BCDE ABCDE Among the participants, Ninja is new to programming and doesn't have much experience; he asks you to solve the problem. Can you help solve this problem? **Sample Input 1:** 2 5 4	Easy

Sample Output 1:

```
E
DE
CDE
BCDE
ABCDE

D
CD
BCD
ABCD
```

Explanation for Sample Input 1:

In the first test case, value of ‘N’ is 5, so print the ‘N’ rows from 1 to ‘N’ where in each row start from (N - i - 1)the character which goes on till ‘Nth character. Hence the answer is [‘E’,’DE’,’CDE,’ BCDE’,’ABCDE’].

In the second test case, the value of ‘N’ is 4, so print the ‘N’ rows from 1 to ‘N’ where each row starts from (N - i - 1)the character, which goes on till ‘Nth character. Hence the answer is [‘D’,’CD’,BCD’,’ABCD’].

Sample Input 2:

```
2
3
2
```

Sample Output 2:

```
C
BC
ABC

B
AB
```

Constraints:

1 <= T <= 50

1 <= N <= 26

3	**Word Pattern** **Problem statement :** Given a pattern and a string s, find if s follows the same pattern. Here follow means a full match, such that there is a bijection between a letter in pattern and a non-empty word in s. **Example 1:** **Input**: pattern = "abba", s = "dog cat cat dog" **Output:** true **Example 2:** **Input:** pattern = "abba", s = "dog cat cat fish" **Output**: false **Example 3:** **Input:** pattern = "aaaa", s = "dog cat cat dog" **Output:** false **Constraints:** 1 <= pattern.length <= 300 pattern contains only lower-case English letters. 1 <= s.length <= 3000 s contains only lowercase English letters and spaces ' '. s does not contain any leading or trailing spaces. All the words in s are separated by a single space.	Easy
4	**Pascal's Triangle II** **Problem statement :** Given an integer rowIndex, return the rowIndexth (0-indexed) row of the Pascal's triangle. In Pascal's triangle, each number is the sum of the two numbers directly above it as shown **Example 1:** **Input:** rowIndex = 3	Easy

	Output: [1,3,3,1] **Example 2:** **Input:** rowIndex = 0 **Output**: [1] **Example 3:** **Input**: rowIndex = 1 **Output**: [1,1] **Constraints:** 0 <= rowIndex <= 33	
5	**Intersection of Two Arrays** **Problem statement :** Given two integer arrays nums1 and nums2, return an array of their intersection Each element in the result must be unique and you may return the result in any order. **Example 1:** Input: nums1 = [1,2,2,1], nums2 = [2,2] Output: [2] **Example 2:** **Input:** nums1 = [4,9,5], nums2 = [9,4,9,8,4] **Output**: [9,4] **Explanation**: [4,9] is also accepted. **Constraints:** 1 <= nums1.length, nums2.length <= 1000 0 <= nums1[i], nums2[i] <= 1000	Easy
6	**Find the Difference of Two Arrays** **Problem statement :** Given two **0-indexed** integer arrays nums1 and nums2, return a list answer of size 2 where:	Easy

	- 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] = []. **Constraints:** - 1 <= nums1.length, nums2.length <= 1000 - -1000 <= nums1[i], nums2[i] <= 1000	
7	**Maximum Product of Two Elements in an Array** **Problem statement :** Given the array of integers nums, you will choose two different indices i and j of that array. Return the maximum value of (nums[i]-1)*(nums[j]-1).	Easy

	Example 1: **Input**: nums = [3,4,5,2] **Output**: 12 **Explanation:** If you choose the indices i=1 and j=2 (indexed from 0), you will get the maximum value, that is, (nums[1]-1)*(nums[2]-1) = (4-1)*(5-1) = 3*4 = 12. **Example 2:** **Input:** nums = [1,5,4,5] **Output**: 16 **Explanation:** Choosing the indices i=1 and j=3 (indexed from 0), you will get the maximum value of (5-1)*(5-1) = 16. **Constraints:** 2 <= nums.length <= 500 1 <= nums[i] <= 10^3	
8	**Degree of an Array** **Problem statement :** Given a non-empty array of non-negative integers nums, the degree of this array is defined as the maximum frequency of any one of its elements. Your task is to find the smallest possible length of a (contiguous) subarray of nums, that has the same degree as nums. **Example 1:** **Input**: nums = [1,2,2,3,1] **Output**: 2 **Explanation:** The input array has a degree of 2 because both elements 1 and 2 appear twice. Of the subarrays that have the same degree: [1, 2, 2, 3, 1], [1, 2, 2, 3], [2, 2, 3, 1], [1, 2, 2], [2, 2, 3], [2, 2] The shortest length is 2. So return 2. **Example 2:** **Input:** nums = [1,2,2,3,1,4,2]	Easy

	Output: 6 **Explanation:** The degree is 3 because the element 2 is repeated 3 times. So [2,2,3,1,4,2] is the shortest subarray, therefore returning 6. **Constraints:** nums.length will be between 1 and 50,000. nums[i] will be an integer between 0 and 49,999.	
9	**Next Greater Element** **Problem statement :** You are given an array ***'a'*** of size ***'n'***. Print the Next Greater Element(NGE) for every element. The Next Greater Element for an element 'x' is the first element on the right side of 'x' in the array, which is greater than 'x' If no greater elements exist to the right of 'x', consider the next greater element as -1. **For example:** **Input:** 'a' = [7, 12, 1, 20] **Output**: NGE = [12, 20, 20, -1] **Explanation**: For the given array, - The next greater element for 7 is 12. - The next greater element for 12 is 20. - The next greater element for 1 is 20. - There is no greater element for 20 on the right side. So we consider NGE as -1. **Sample Input 1:** 5 1 5 3 4 2 **Sample Output 1:** 5 -1 4 -1 -1 **Sample Input 2:** 5	Easy

	5 5 5 5 5 **Sample Output 2:** -1 -1 -1 -1 -1 **Constraints :** 1 <= 'n' <= 10^5 1 <= 'a[i]' <= 10^9	
10	**Single Element in a Sorted Array** **Problem statement :** You are given a sorted array ***'arr'*** of ***'n'*** numbers such that every number occurred twice in the array except one, which appears only once. Return the number that appears once. **Example:** **Input:** 'arr' = [1,1,2,2,4,5,5] **Output**: 4 **Explanation:** Number 4 only appears once the array. **Note :** Exactly one number in the array 'arr' appears once. **Sample Input 1 :** 5 1 1 3 5 5 **Sample Output 1 :** 3 **Explanation of Sample Input 1 :** Given array is [1, 1, 3, 5, 5] Here, 3 occurs once in the array. So, the answer is 3. **Sample Input 2 :** 5 1 1 4 4 15 **Sample Output 2 :**	Easy

15

Explanation of Sample Input 2 :

The array is [1, 1, 4, 4, 15].

Here, 15 occurs once in the array. So, the answer is 15.

Constraints :

1 <= n <= 10^5

0 <= arr[i] <= 10^9

11 Pair Sum

Problem statement

You are given an integer array 'ARR' of size 'N' and an integer 'S'. Your task is to return the list of all pairs of elements such that each sum of elements of each pair equals 'S'.

Note:

Each pair should be sorted i.e the first value should be less than or equals to the second value.

Return the list of pairs sorted in non-decreasing order of their first value. In case if two pairs have the same first value, the pair with a smaller second value should come first.

Sample Input 1:

5 5

1 2 3 4 5

Sample Output 1:

1 4

2 3

Explaination For Sample Output 1:

Here, 1 + 4 = 5

2 + 3 = 5

Hence the output will be, (1,4) , (2,3).

Sample Input 2:

5 0

2 -3 3 3 -2

Sample Output 2:

-3 3

-3 3

-2 2

Constraints:

1 <= N <= 10^3

-10^5 <= ARR[i] <= 10^5

-2 * 10^5 <= S <= 2 * 10^5

12 Summary Ranges

Problem statement : You are given a **sorted unique** integer array nums.

A **range** [a,b] is the set of all integers from a to b (inclusive).

Return *the* ***smallest sorted*** *list of ranges that* ***cover all the numbers in the array exactly***. That is, each element of nums is covered by exactly one of the ranges, and there is no integer x such that x is in one of the ranges but not in nums.

Each range [a,b] in the list should be output as:

- "a->b" if a != b
- "a" if a == b

Example 1:

Input: nums = [0,1,2,4,5,7]

Output: ["0->2","4->5","7"]

Explanation: The ranges are:

[0,2] --> "0->2"

[4,5] --> "4->5"

[7,7] --> "7"

Example 2:

Input: nums = [0,2,3,4,6,8,9]

Output: ["0","2->4","6","8->9"]

Explanation: The ranges are:

[0,0] --> "0"

[2,4] --> "2->4"

[6,6] --> "6"

[8,9] --> "8->9"

Constraints:

- 0 <= nums.length <= 20
- -2^{31} <= nums[i] <= 2^{31} - 1
- All the values of nums are **unique**.
- nums is sorted in ascending order.

13 **Kids With the Greatest Number of Candies**

Problem statement : 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 ith 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

14 **Set Mismatch**

Problem statement : You have a set of integers s, which originally contains all the numbers from 1 to n. Unfortunately, due to some error, one of the numbers in s got duplicated to another number in the set, which results in repetition of one number and loss of another number.

You are given an integer array nums representing the data status of this set after the error.

Find the number that occurs twice and the number that is missing and return them in the form of an array.

Example 1:

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

Output: [2,3]

Example 2:

Input: nums = [1,1]

Output: [1,2]

Constraints:

2 <= nums.length <= 10^4

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

15 The K Weakest Rows in a Matrix

Problem statement: You are given an m x n binary matrix mat of 1's (representing soldiers) and 0's (representing civilians). The soldiers are positioned **in front** of the civilians. That is, all the 1's will appear to the **left** of all the 0's in each row. A row i is **weaker** than a row j if one of the following is true:

- The number of soldiers in row i is less than the number of soldiers in row j.
- Both rows have the same number of soldiers and i < j.
  Return the indices of the k **weakest** rows in the matrix ordered from weakest to strongest.

Example 1:

Input: mat =

[[1,1,0,0,0],

[1,1,1,1,0],

[1,0,0,0,0],

[1,1,0,0,0],

[1,1,1,1,1]],

k = 3

Output: [2,0,3]

Explanation:

The number of soldiers in each row is:

- Row 0: 2

- Row 1: 4

- Row 2: 1

- Row 3: 2

- Row 4: 5

The rows ordered from weakest to strongest are [2,0,3,1,4].

Example 2:

Input: mat =

[[1,0,0,0],

[1,1,1,1],

[1,0,0,0],

[1,0,0,0]],

k = 2

Output: [0,2]

Explanation:

The number of soldiers in each row is:

- Row 0: 1

- Row 1: 4

- Row 2: 1

- Row 3: 1

The rows ordered from weakest to strongest are [0,2,3,1].

Constraints:

- m == mat.length
- n == mat[i].length
- 2 <= n, m <= 100
- 1 <= k <= m

matrix[i][j] is either 0 or 1.

16 **Lucky Numbers in Matrix**

Problem statement : Given an m x n matrix of distinct numbers, return all lucky numbers in the matrix in any order.

A lucky number is an element of the matrix such that it is the minimum element in its row and maximum in its column.

Example 1:

Input: matrix = [[3,7,8],[9,11,13],[15,16,17]]

Output: [15]

Explanation: 15 is the only lucky number since it is the minimum in its row and the maximum in its column.

Example 2:

Input: matrix = [[1,10,4,2],[9,3,8,7],[15,16,17,12]]

Output: [12]

Explanation: 12 is the only lucky number since it is the minimum in its row and the maximum in its column.

Example 3:

Input: matrix = [[7,8],[1,2]]

Output: [7]

Explanation: 7 is the only lucky number since it is the minimum in its row and the maximum in its column.

Constraints:

m == mat.length

n == mat[i].length

1 <= n, m <= 50

1 <= matrix[i][j] <= 105.

All elements in the matrix are distinct.

17 **Destination City**

Problem statement : You are given the array paths, where paths[i] = [cityA_i, cityB_i] means there exists a direct path going from cityA_i to cityB_i. Return the destination city, that is, the city without any path outgoing to another city.

It is guaranteed that the graph of paths forms a line without any loop, therefore, there will be exactly one destination city.

Example 1:

Input: paths = [["London","New York"],["New York","Lima"],["Lima","Sao Paulo"]]

Output: "Sao Paulo"

Explanation: Starting at "London" city you will reach "Sao Paulo" city which is the destination city. Your trip consist of: "London" -> "New York" -> "Lima" -> "Sao Paulo".

Example 2:

Input: paths = [["B","C"],["D","B"],["C","A"]]

Output: "A"

Explanation: All possible trips are:

"D" -> "B" -> "C" -> "A".

"B" -> "C" -> "A".

"C" -> "A".

"A".

Clearly the destination city is "A".

Example 3:

Input: paths = [["A","Z"]]

Output: "Z"

Constraints:

- 1 <= paths.length <= 100
- paths[i].length == 2
- 1 <= $cityA_i$.length, $cityB_i$.length <= 10
- $cityA_i$!= $cityB_i$
- All strings consist of lowercase and uppercase English letters and the space character.

18 Check if a String Is an Acronym of Words

Problem statement: Given an array of strings words and a string s, determine if s is an acronym of words.

The string s is considered an acronym of words if it can be formed by concatenating the first character of each string in words in order. For example, "ab" can be formed from ["apple", "banana"], but it can't be formed from ["bear", "aardvark"].

Return true if s is an acronym of words, and false otherwise.

Example 1:

Input: words = ["alice","bob","charlie"], s = "abc"

Output: true

Explanation: The first character in the words "alice", "bob", and "charlie" are 'a', 'b', and 'c', respectively. Hence, s = "abc" is the acronym.

Example 2:

Input: words = ["an","apple"], s = "a"

Output: false

Explanation: The first character in the words "an" and "apple" are 'a' and 'a', respectively.

The acronym formed by concatenating these characters is "aa".

Hence, s = "a" is not the acronym.

Example 3:

Input: words = ["never","gonna","give","up","on","you"], s = "ngguoy"

Output: true

Explanation: By concatenating the first character of the words in the array, we get the string "ngguoy".

Hence, s = "ngguoy" is the acronym.

Constraints:

1 <= words.length <= 100

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

1 <= s.length <= 100

words[i] and s consist of lowercase English letters.

19 Find Words That Can Be Formed by Characters

Problem statement : You are given an array of strings words and a string chars.

A string is good if it can be formed by characters from chars (each character can only be used once).

Return the sum of lengths of all good strings in words.

Example 1:

Input: words = ["cat","bt","hat","tree"], chars = "atach"

Output: 6

Explanation: The strings that can be formed are "cat" and "hat" so the answer is 3 + 3 = 6.

Example 2:

Input: words = ["hello","world","leetcode"], chars = "welldonehoneyr"

Output: 10

Explanation: The strings that can be formed are "hello" and "world" so the answer is 5 + 5 = 10.

Constraints:

1 <= words.length <= 1000

1 <= words[i].length, chars.length <= 100

words[i] and chars consist of lowercase English letters.

20 Redistribute Characters to Make All Strings Equal

Problem statement : You are given an array of strings words (0-indexed).

In one operation, pick two distinct indices i and j, where words[i] is a non-empty string, and move any character from words[i] to any position in words[j].

Return true if you can make every string in words equal using any number of operations, and false otherwise.

Example 1:

Input: words = ["abc","aabc","bc"]

Output: true

Explanation: Move the first 'a' in words[1] to the front of words[2], to make words[1] = "abc" and words[2] = "abc".

All the strings are now equal to "abc", so return true.

Example 2:

Input: words = ["ab","a"]

Output: false

Explanation: It is impossible to make all the strings equal using the operation.

Constraints:

1 <= words.length <= 100

1 <= words[i].length <= 100

words[i] consists of lowercase English letters.

21 First Unique Character in a String

Problem statement : Given a string s, find the first non-repeating character in it and return its index. If it does not exist, return -1.

Example 1:

Input: s = "leetcode"

Output: 0

Example 2:

Input: s = "loveleetcode"

Output: 2

Example 3:

Input: s = "aabb"

Output: -1

Constraints:

1 <= s.length <= 10^5

s consists of only lowercase English letters.

22 Ransom Note

Problem statement : Given two strings ransomNote and magazine, return true if ransomNote can be constructed by using the letters from magazine and false otherwise.

Each letter in magazine can only be used once in ransomNote.

Example 1:

Input: ransomNote = "a", magazine = "b"

Output: false

Example 2:

Input: ransomNote = "aa", magazine = "ab"

Output: false

Example 3:

Input: ransomNote = "aa", magazine = "aab"

Output: true

Constraints:

1 <= ransomNote.length, magazine.length <= 10^5

ransomNote and magazine consist of lowercase English letters.

23 Unique Morse Code Words

Problem statement : 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:

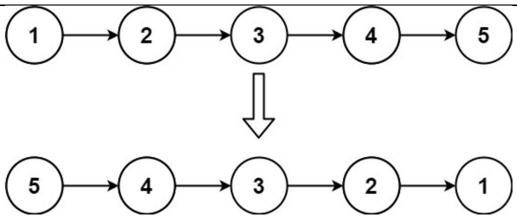
Input: words = ["a"]

Output: 1

Constraints:

- 1 <= words.length <= 100
- 1 <= words[i].length <= 12

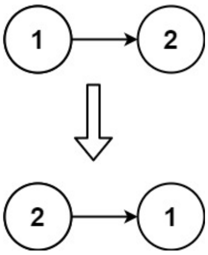
	words[i] consists of lowercase English letters.	
24	**Valid Parentheses** **Problem statement :** You're given a string ***'S'*** consisting of "{", "}", "(", ")", "[" and "]" . Return true if the given string 'S' is balanced, else return false. **For example:** 'S' = "{}()". There is always an opening brace before a closing brace i.e. '{' before '}', '(' before '). So the 'S' is Balanced. **Sample Input 1 :** [()]{}{[()()]()} **Sample Output 1 :** Balanced **Explanation Of the Sample Input 1 :** There is always an opening brace before a closing brace i.e. '{' before '}', '(' before '), '[' before ']'. So the 'S' is Balanced. **Sample Input 2 :** [[}[**Sample Output 2 :** Not Balanced **Constraints:** 1 <= 'N' <= 10^5 Where 'N' is the length of the input string 'S'.	
25	**Reverse Linked List** **Problem statement :** 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: []

Constraints:

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

-5000 <= Node.val <= 5000

26 **Palindrome Linked List**

Problem statement : You are given a singly Linked List of integers. Your task is to return true if the given singly linked list is a palindrome otherwise returns false.

For example:

The given linked list is 1 -> 2 -> 3 -> 2-> 1-> NULL.

It is a palindrome linked list because the given linked list has the same order of elements when traversed forwards and backward.

Sample Input 1 :

2

1 2 3 4 5 6 -1

1 2 1 -1

Sample Output 1 :

false

true

Explanation for sample 1:

For the first test case, it is not a palindrome because Linked List doesn't have the same order of elements when traversed forwards and backwards.

For the second test case, it is a palindrome linked list because a Linked List has the same order of elements when traversed forwards and backwards.

Sample Input 2 :

2

1 -1

1 10 45 10 1 -1

Sample Output 2 :

true

True

Constraints :

1 <= T <= 10

0 <= L <= 10^5

1 <= data <= 10^9 and data != -1

Where L is the number of nodes in the Linked List.

27 Reverse First K elements of Queue

Problem statement : You are given a QUEUE containing ‘N’ integers and an integer ‘K’. You need to reverse the order of the first ‘K’ elements of the queue, leaving the other elements in the same relative order.

You can only use the standard operations of the QUEUE STL:

1. enqueue(x) : Adds an item x to rear of the queue
2. dequeue() : Removes an item from front of the queue
3. size() : Returns number of elements in the queue.
4. front() : Finds the front element.

For Example:

Let the given queue be { 1, 2, 3, 4, 5 } and K be 3.

You need to reverse the first K integers of Queue which are 1, 2, and 3.

Thus, the final response will be { 3, 2, 1, 4, 5 }.

Sample Input 1:

```
2
5 3
1 2 3 4 5
4 2
6 2 4 1
```

Sample Output 1:

```
3 2 1 4 5
2 6 4 1
```

Explanation:

For test case 1: Refer to the example explained above.

For test case 2: The queue after reversing the first 2 elements i.e., 6 and 2 will be { 2, 6, 4, 1 }.

Sample Input 2:

```
2
5 2
5 3 2 6 4
4 4
1 2 3 4
```

Sample Output 2:

```
3 5 2 6 4
4 3 2 1
```

Constraints:

1 <= T <= 10

1 <= N <= 10 ^ 5

0 <= K <= N

-10 ^ 9 <= queue elements <= 10 ^ 9

28 **Delete middle element from stack**

Problem statement : You are having a stack "ARR" of size 'N+1', your task is to delete the middlemost element so that the size of resulting stack is 'N'.

A stack is a linear data structure where both insertion and deletion of elements take place at the top. It follows FILO (First In Last Out) or LIFO (Last In First Out) approaches. Books piled on top of each other is an example of a stack, where you can only remove a single book at a time, which is at the top of the stack. Likewise, you can only add a single book at a time, on the top of the stack only.

Example :-

INPUT : ARR [ ] = [1 , 2 , 3 , 4 , 5] , N = 4

OUTPUT: ARR [ ] = [1 , 2 , 4, 5]

The above example contains an odd number of elements, hence the middle element is clearly the (N+1) / 2th element, which is removed from the stack in the output.

INPUT : ARR [ ] = [5, 6, 7, 8] , N = 3

OUTPUT: ARR [ ] = [5, 7, 8]

The above example contains an even number of elements, so out of the two middle elements, we consider the one which occurs first. Hence, the middle element would be ((N+1) / 2 - 1) element, which is 6 and is removed from the stack in the output.

Sample Input 1:

```
2
4
1 2 3 4 5
7
83 74 67 49 94 8 11 1
```

Sample Output 1:

```
1 2 4 5
83 74 67 94 8 11 1
```

Explanation for Sample 1:

In the 1st testcase, there are an odd number of elements, hence the middle element is clearly the (N+1) / 2th element which is 3, and is removed from the stack in the output.

In the 2nd testcase, there are an odd number of elements, hence the middle element is clearly the (N+1) / 2th element which is 49, and is removed from the stack in the output.

Sample Input 2:

```
3
1
5 10
4
1 3 4 2 7
5
9 5 2 7 8 6
```

Sample Output 2:

```
10
1 3 2 7
9 5 7 8 6
```

Constraints:

1 <= T <= 100

1 <= N+1 <= 3000

0 <= data <= 10^9

Where ‘T’ is the number of test cases, ‘N+1’ is the number of elements in the input Stack. ‘data’ is the value of each element in the stack.

29 **Flatten A Linked List**

Problem statement : You are given a linked list containing ***'n' 'head'*** nodes, where every node in the linked list contains two pointers:

(1) ***‘next’*** which points to the next node in the list

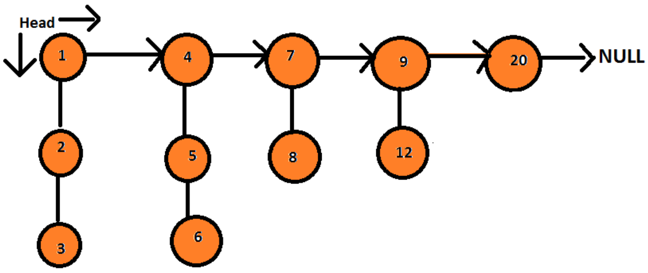
(2) ***‘child’*** pointer to a linked list where the current node is the head.

Each of these child linked lists is in sorted order and connected by 'child' pointer.

Your task is to flatten this linked such that all nodes appear in a single layer or level in a ***'sorted order'.***

Example:

Input: Given linked list is:



Output:

1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 12 → 20 → null

Explanation:

The returned linked list should be in a sorted order. All the elements in this returned linked list are connected by 'child' pointers and 'next' pointers point to null.

Sample Input 1 :

4

3

1 2 3

3

8 10 15

2

18 22

1

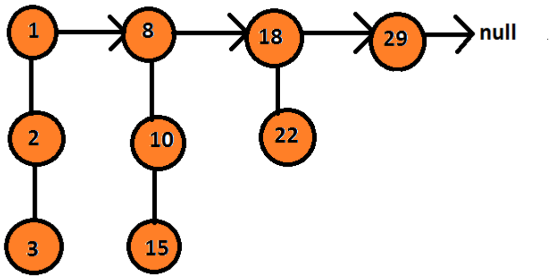
29

Sample Output 1 :

1 2 3 8 10 15 18 22 29

Explanation For Sample Input 1:

The given linked list is



Therefore after flattening the list will become-

1 -> 2 -> 3 -> 8 -> 10 -> 15 -> 18 -> 22 -> 29 ->null

Sample Input 2 :

5

2

4 6

2

5 71

3

7 8 9

3

11 12 19

3

14 15 17

Sample Output 2 :

4 5 6 7 8 9 11 12 14 15 17 19 71

Constraints:

1 <= n <= 100

1 <= k <= 20

1 <= Node.data <= 10^9

30 **Middle Of Linked List**

Problem statement : Given a singly linked list of ***'N'*** nodes. The objective is to determine the middle node of a singly linked list. However, if the list has an even number of nodes, we return the second middle node.

Sample Input 1 :

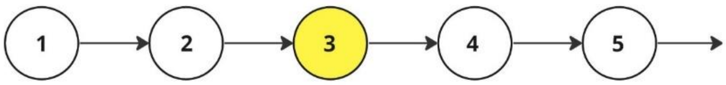
5

1 2 3 4 5

Sample Output 1 :

3 4 5

Explanation Of Sample Input 1 :



We can clearly see that there are 5 elements in the linked list therefore the middle node is the node with value '3'.

Sample Input 2 :

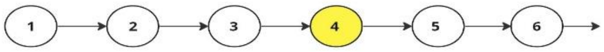
6

1 2 3 4 5 6

Sample Output 2 :

4 5 6

Explanation Of Sample Input 2 :



We can clearly see that there are 6 elements in the linked list and the middle nodes are nodes with values 3 and 4 hence we return a second middle node having value '4'.

Constraints :

1 <= 'N' <= 10^4

0 <= 'data' <= 10^3

Where 'N' is the length of the linked list.