



**Self-Practice 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 Details	Level
1	<p><b>Game of Numbers</b></p> <p><b>Problem statement :</b> Tom loves playing with numbers. So one day, he wants to arrange a few numbers in the 'N' number of rows. The first row contains 1 number, the second row has two numbers, and so on.</p> <p>On row 1, he places '1'. From the second row, he puts a number equal to one less than the number of the row at two ends of the row and places zeros in between.</p> <p>You are given an integer 'N' denoting the given number of rows. Can you print the pattern Tom wants to create?</p> <p>Pattern for N = 4:</p> <pre>1 11 202 3003</pre> <p><b>Sample Input 1 :</b></p> <pre>2 3 1</pre> <p><b>Sample Output 1 :</b></p> <pre>1 11 202 1</pre> <p><b>Explanation for Sample Input 1 :</b></p>	Easy

*Little practice is worth more than a ton of theory*



	<p>In the first test case, we are required to create a pattern consisting of 3 lines. First-line contains '1'. From the second line, we have to place a number one less than the row number at two ends of the row and place zeros in between. So we place '1' at two ends, and since the second line contains only two numbers, so there will be no '0'. In the third line, we have to place '2', because row number is 3 and one number less than that is '2', at two ends. This line will have 3 digits, so we have to place one '0' between the 2 '2'.</p> <p>In the second test case, we are required to create a pattern consisting of 1 line. The first and only line will have '1'.</p> <p><b>Sample Input 2 :</b></p> <pre>2 8 2</pre> <p><b>Sample Output 2 :</b></p> <pre>1 11 202 3003 40004 500005 6000006 70000007 1 11</pre> <p><b>Constraints:</b></p> <pre>1 &lt;= T &lt;= 10 1 &lt;= N &lt;= 10^2</pre> <p>Where 'T' denotes number of test cases and 'N' is the given integer denoting the given number of rows.</p>	
<b>2</b>	<b>Funny Divisors</b> <b>Problem statement</b>	



Ninja is bored with his previous game of numbers, so now he is playing with divisors.

He is given 'N' numbers, and his task is to return the sum of all numbers which is divisible by 2 or 3.

Let the number given to him be - 1, 2, 3, 5, 6. As 2, 3, and 6 is divisible by either 2 or 3 we return  $2 + 3 + 6 = 11$ .

**Sample Input 1 :**

2

3

1 2 3

4

5 6 9 8

**Sample Output 1 :**

5

23

**Explanation for Sample Input 1 :**

In the first test case, 1 is neither divisible by 2 or 3. 2 is divisible by 2, and 3 is divisible by 3. So here we return the sum of 2 + 3 which is equal to 5.

In the second test case, 5 is divisible by neither 5 nor 6. 6 is divisible by 2, 9 is divisible by 3, and 8 is divisible by 2. So here we return  $6 + 9 + 8 = 23$ .

**Sample Input 2 :**

2

7

7 5 11 3 5 2 9

2

3 4

**Sample Output 2 :**

14

7

**Constraints:**

$1 \leq T \leq 10$

*Little practice is worth more than a ton of theory*



	$1 \leq N \leq 10^3$ $0 \leq \text{input}[i] \leq 10^3$ <p>Where 'T' denotes the number of test cases and 'N' is the elements given to Ninja and input[i] denotes theith input.</p>	
<b>3</b>	<b>Maximum Frequency Number</b> <p><b>Problem statement :</b> Helen is given an array of integers that contain numbers in random order. He needs to write a program to find and return the number which occurs the maximum times in the given input. He needs your help to solve this problem. If two or more elements contend for the maximum frequency, return the element which occurs in the array first i.e. whose index is lowest.</p> <p>For example,  For 'arr' = [ 1, 2, 3, 1, 2]. you need to return 1.</p> <p><b>Sample Input 1:</b></p> <pre>1 13 2 12 2 11 -12 2 -1 2 2 11 12 2 -6</pre> <p><b>Sample Output 1:</b></p> <pre>2</pre> <p><b>Explanation of Sample Input 1:</b></p> <p>Test case 1:  For the first test case of sample output 1, as we start traveling the array, '2' has the highest frequency, hence our answer is '2'.</p> <p><b>Sample Input 2:</b></p> <pre>2 3 4 -5 1 4 1 -2 1 -2</pre> <p><b>Sample Output 2:</b></p> <pre>4</pre>	

***Little practice is worth more than a ton of theory***



	<p>-2</p> <p><b>Explanation of Sample Input 2:</b></p> <p>Test case 1: For the first test case of sample output 2, as all the elements have only occurred once, so we will take the first element that has occurred once. Therefore our output will be '4'.</p> <p>Test case 2: -2 has the highest frequency.</p> <p><b>Constraints:</b></p> <p><math>1 \leq T \leq 5</math></p> <p><math>1 \leq N \leq 10000</math></p> <p><math>-10^3 \leq  arr  \leq 10^3</math></p>	
4	<p><b>Make Array Zero by Subtracting Equal Amounts</b></p> <p><b>Problem statement :</b> You are given a non-negative integer array nums. In one operation, you must: Choose a positive integer x such that x is less than or equal to the smallest non-zero element in nums. Subtract x from every positive element in nums. Return the minimum number of operations to make every element in nums equal to 0.</p> <p><b>Example 1:</b> <b>Input:</b> nums = [1,5,0,3,5] <b>Output:</b> 3 <b>Explanation:</b> In the first operation, choose x = 1. Now, nums = [0,4,0,2,4]. In the second operation, choose x = 2. Now, nums = [0,2,0,0,2]. In the third operation, choose x = 2. Now, nums = [0,0,0,0,0].</p> <p><b>Example 2:</b> <b>Input:</b> nums = [0] <b>Output:</b> 0 <b>Explanation:</b> Each element in nums is already 0 so no operations are needed.</p>	

***Little practice is worth more than a ton of theory***



	<p><b>Constraints:</b></p> <p><math>1 \leq \text{nums.length} \leq 100</math></p> <p><math>0 \leq \text{nums}[i] \leq 100</math></p>	
<b>5</b>	<p><b>Last Stone Weight</b></p> <p><b>Problem statement :</b> You are given an array of integers stones where stones[i] is the weight of the ith stone.</p> <p>We are playing a game with the stones. On each turn, we choose the heaviest two stones and smash them together. Suppose the heaviest two stones have weights x and y with <math>x \leq y</math>. The result of this smash is:</p> <p>If <math>x == y</math>, both stones are destroyed, and</p> <p>If <math>x \neq y</math>, the stone of weight x is destroyed, and the stone of weight y has new weight <math>y - x</math>.</p> <p>At the end of the game, there is at most one stone left. Return the weight of the last remaining stone. If there are no stones left, return 0.</p> <p><b>Example 1:</b></p> <p><b>Input:</b> stones = [2,7,4,1,8,1]</p> <p><b>Output:</b> 1</p> <p><b>Explanation:</b></p> <p>We combine 7 and 8 to get 1 so the array converts to [2,4,1,1,1] then,</p> <p>we combine 2 and 4 to get 2 so the array converts to [2,1,1,1] then,</p> <p>we combine 2 and 1 to get 1 so the array converts to [1,1,1] then,</p> <p>we combine 1 and 1 to get 0 so the array converts to [1] then that's the value of the last stone.</p> <p><b>Example 2:</b></p> <p><b>Input:</b> stones = [1]</p> <p><b>Output:</b> 1</p>	

*Little practice is worth more than a ton of theory*



	<p><b>Constraints:</b></p> <p><math>1 \leq \text{stones.length} \leq 30</math></p> <p><math>1 \leq \text{stones}[i] \leq 1000</math></p>	
<b>6</b>	<p><b>Kth Largest Element in a Stream</b></p> <p><b>Problem statement :</b> Design a class to find the kth largest element in a stream. Note that it is the kth largest element in the sorted order, not the kth distinct element.</p> <p>Implement KthLargest class:</p> <p>KthLargest(int k, int[] nums) Initializes the object with the integer k and the stream of integers nums.</p> <p>int add(int val) Appends the integer val to the stream and returns the element representing the kth largest element in the stream.</p> <p><b>Example 1:</b></p> <p><b>Input</b></p> <p>["KthLargest", "add", "add", "add", "add", "add"]</p> <p>[[3, [4, 5, 8, 2]], [3], [5], [10], [9], [4]]</p> <p><b>Output</b></p> <p>[null, 4, 5, 5, 8, 8]</p> <p><b>Explanation</b></p> <pre>KthLargest kthLargest = new KthLargest(3, [4, 5, 8, 2]); kthLargest.add(3); // return 4 kthLargest.add(5); // return 5 kthLargest.add(10); // return 5 kthLargest.add(9); // return 8 kthLargest.add(4); // return 8</pre> <p><b>Constraints:</b></p> <p><math>1 \leq k \leq 10^4</math></p> <p><math>0 \leq \text{nums.length} \leq 10^4</math></p> <p><math>-10^4 \leq \text{nums}[i] \leq 10^4</math></p> <p><math>-10^4 \leq \text{val} \leq 10^4</math></p>	

***Little practice is worth more than a ton of theory***



	<p>At most <math>10^4</math> calls will be made to add.</p> <p>It is guaranteed that there will be at least <math>k</math> elements in the array when you search for the <math>k</math>th element.</p>	
<b>7</b>	<p><b>Two Sum</b></p> <p><b>Problem statement :</b> You are given an array of integers 'ARR' of length 'N' and an integer Target. Your task is to return all pairs of elements such that they add up to Target.</p> <p><b>Note:</b> We cannot use the element at a given index twice.</p> <p><b>Sample Input 1 :</b> 2 4 9 2 7 11 13 5 1 1 -1 -1 2 2</p> <p><b>Sample Output 1:</b> 2 7 -1 2 -1 2</p> <p><b>Explanation for Sample 1:</b> For the first test case, we can see that the sum of 2 and 7 is equal to 9 and it is the only valid pair. For the second test case, there are two valid pairs (-1,2) and (-1,2), which add up to 1.</p> <p><b>Sample Input 2 :</b> 1 4 16 2 7 11 13</p> <p><b>Sample Output 2 :</b> -1 -1</p> <p><b>Constraints:</b> <math>1 \leq n \leq 10^5</math></p>	

*Little practice is worth more than a ton of theory*

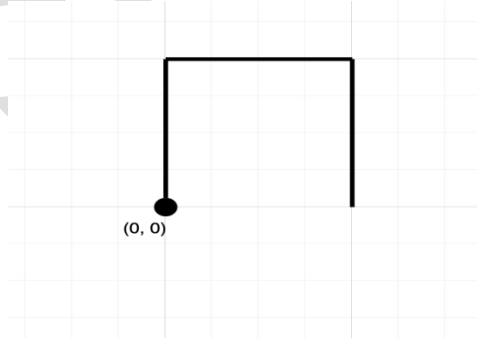




	$1 \leq \text{arr}[i] \leq 10^5$  Where 'T' denotes the number of test cases, 'N' represents the size of the array, 'TARGET' represents the sum required, and 'ARR[i]' represents array elements.	
<b>8</b>	<b>Sort a Stack</b>  <b>Problem statement :</b> You're given a stack consisting of 'N' integers. Your task is to sort this stack in descending order using recursion.  We can only use the following functions on this stack S. is_empty(S) : Tests whether stack is empty or not. push(S) : Adds a new element to the stack. pop(S) : Removes top element from the stack. top(S) : Returns value of the top element. Note that this function does not remove elements from the stack.  <b>Note :</b> 1) Use of any loop constructs like while, for..etc is not allowed. 2) The stack may contain duplicate integers. 3) The stack may contain any integer i.e it may either be negative, positive or zero.  <b>Sample Input 1:</b> 1 5 5 -2 9 -7 3 <b>Sample Output 1:</b> 9 5 3 -2 -7  <b>Explanation of Sample Input 1:</b> 9 Is the largest element, hence it's present at the top. Similarly $5 > 3$ , $3 > -2$ and -7 being the smallest element is present at the last.  <b>Sample Input 2:</b>	

*Little practice is worth more than a ton of theory*



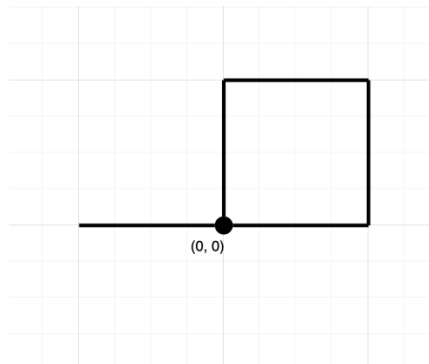
	<p>1 5 -3 14 18 -5 30</p> <p><b>Sample Output 2:</b> 30 18 14 -3 -5</p> <p><b>Explanation of Sample Input 2:</b> 30 is the largest element, hence it's present at the top. Similarly, <math>18 &gt; 14</math>, <math>14 &gt; -3</math> and -5 being the smallest element is present at the last.</p> <p><b>Constraints:</b> <math>1 \leq 'T' \leq 100</math> <math>1 \leq 'N' \leq 100</math> <math>1 \leq ' V  \leq 10^9</math></p> <p>Where <math> V </math> denotes the absolute value of any stack element.</p>	
<b>9</b>	<p><b>Path Crossing</b></p> <p><b>Problem statement :</b> Given a string path, where <math>\text{path}[i] = \text{'N', 'S', 'E' or 'W'}</math>, each representing moving one unit north, south, east, or west, respectively. You start at the origin (0, 0) on a 2D plane and walk on the path specified by path. Return true if the path crosses itself at any point, that is, if at any time you are on a location you have previously visited. Return false otherwise.</p> <p><b>Example 1:</b></p>  <p><b>Input:</b> path = "NES" <b>Output:</b> false</p>	

*Little practice is worth more than a ton of theory*



**Explanation:** Notice that the path doesn't cross any point more than once.

**Example 2:**



**Input:** path = "NESWW"

**Output:** true

**Explanation:** Notice that the path visits the origin twice.

**Constraints:**

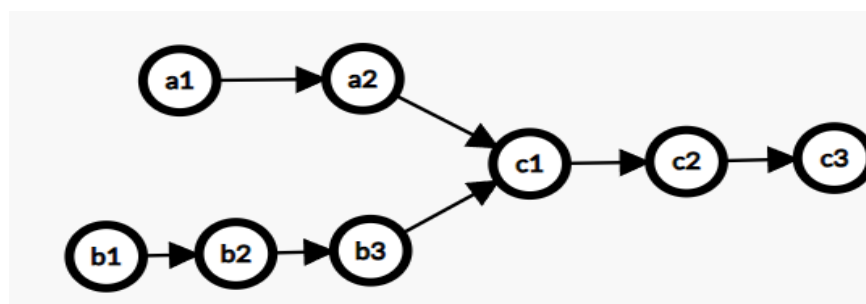
- $1 \leq \text{path.length} \leq 10^4$
- path[i] is either 'N', 'S', 'E', or 'W'.

## 10 Intersection of Two Linked Lists

**Problem statement :** You are given two Singly Linked Lists of integers, which may have an intersection point. Your task is to return the first intersection node. If there is no intersection, return NULL.

**Example:-**

The Linked Lists, where a1, a2, c1, c2, c3 is the first linked list and b1, b2, b3, c1, c2, c3 is the second linked list, merging at node c1.



*Little practice is worth more than a ton of theory*



**Sample Input 1 :**

4 1 -1

5 6 -1

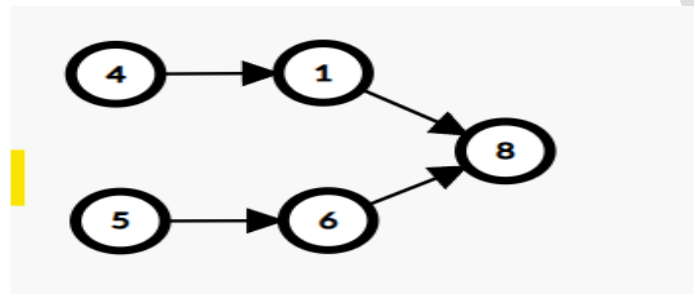
8 -1

**Sample Output 1 :**

8

**Explanation For Sample Input 1:**

As shown in the diagram, the node with data is 8, at which merging starts



**Sample Input 2 :**

1 9 1 -1

3 -1

2 -1

**Sample Output 2 :**

2

**Constraints :**

$-10^9 \leq \text{data} \leq 10^9$  and  $\text{data} \neq -1$

The length of both the linked list is positive.