

**Exercise No. : 6****Topics Covered : Array, String, Arraylist, Priority Queue, Control Flow Statements, Bit Manipulation, Treemap, Hashset****Date : 21-05-2024****Solve the following problems**

Q No.	Question Detail	Level
1	<p><b>Diamond of numbers</b></p> <p><b>Problem Statement:</b> Students at Ninja School were bored with their daily routine and wanted to try something new. So, they thought about observing patterns. While observing the patterns, they found a very interesting pattern which they wished to explore. Your task is to help the students to visualize the pattern for any given 'N'. Given an integer 'N' you need to return the corresponding pattern for it.</p> <p>For example, pattern for N = 15 will be:</p> <pre>1 123 12345 1234567 123456789 12345678912 1234567891234 123456789123456 1234567891234 12345678912 123456789 1234567 12345 123 1</pre>	Medium

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

**Sample Input 1:**

2

7

4

**Sample Output 1:**

1

123

12345

1234567

12345

123

1

1

123

12345

123

1

**Sample Input 2:**

2

5

6

**Sample Output 2:**

1

123

12345

123

1

1

123

12345

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



	<p>1234567 12345 123 1</p> <p><b>Constraints:</b></p> <p><math>1 \leq T \leq 10</math> <math>1 \leq N \leq 500</math></p>	
<b>2</b>	<p><b>Subarray Sum Equals K</b></p> <p><b>Problem Statement:</b> Given an array of integers nums and an integer k, return the total number of subarrays whose sum equals to k. A subarray is a contiguous non-empty sequence of elements within an array.</p> <p><b>Example 1:</b> Input: nums = [1,1,1], k = 2 Output: 2</p> <p><b>Example 2:</b> Input: nums = [1,2,3], k = 3 Output: 2</p> <p><b>Constraints:</b></p> <p><math>1 \leq \text{nums.length} \leq 2 * 10^4</math> <math>-1000 \leq \text{nums}[i] \leq 1000</math> <math>-10^7 \leq k \leq 10^7</math></p>	Medium
<b>3</b>	<p><b>Longest Subarray With Sum K.</b></p> <p><b>Problem Statement:</b> Alex and his friend are playing a game of subarrays. They have an array 'NUMS' of length 'N'. Alex's friend gives him an arbitrary integer 'K' and asks him to find the length of the longest subarray in which the sum of elements is equal to 'K'.</p>	Medium

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



Alex asks for your help to win this game. Find the length of the longest subarray in which the sum of elements is equal to 'K'.

If there is no subarray whose sum is 'K' then you should return 0.

**Sample Input 1 :**

```
2
3 5
2 3 5
3 1
-1 1 1
```

**Sample Output 1:**

```
2
3
```

**Explanation Of Sample Input 1:**

For the first case:

There are two subarrays with sum = 5, [2, 3] and [5]. Hence the length of the longest subarray is 2.

For the second case:

Longest subarray with sum = 1 is [1, -1, 1].

**Sample Input 2:**

```
2
3 4
1 1 1
3 2
-50 0 52
```

**Sample Output 2:**

```
0
3
```

**Constraints:**

$1 \leq T \leq 10$

$1 \leq N \leq 10^5$

$-10^6 \leq \text{NUMS}[i] \leq 10^6$

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



	$-10^6 \leq K \leq 10^6$	
4	<p><b>Top K Frequent Elements</b></p> <p><b>Problem Statement:</b> Given an integer array nums and an integer k, return the k most frequent elements. You may return the answer in any order.</p> <p><b>Example 1:</b> <b>Input:</b> nums = [1,1,1,2,2,3], k = 2 <b>Output:</b> [1,2]</p> <p><b>Example 2:</b> <b>Input:</b> nums = [1], k = 1 <b>Output:</b> [1]</p> <p><b>Constraints:</b></p> <p>1 ≤ nums.length ≤ 105 -104 ≤ nums[i] ≤ 104 k is in the range [1, the number of unique elements in the array]. It is guaranteed that the answer is unique.</p>	Medium
5	<p><b>K Largest Element</b></p> <p><b>Problem Statement:</b> You are given an unsorted array containing 'N' integers. You need to find 'K' largest elements from the given array. Also, you need to return the elements in non-decreasing order.</p> <p><b>Sample Input 1:</b></p> <p>2 4 2 3 4 2 1 5 1 2 2 3 3 1</p> <p><b>Sample Output 1:</b></p> <p>3 4 3</p>	Medium

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



	<p><b>Explanation for sample input 1:</b></p> <p><b>Test case 1:</b> If we sort the array then it will look like: [1, 2, 3, 4]. The 2 largest elements will be [3, 4].</p> <p><b>Test case 2:</b> If we sort the array then it will look like: [1, 2, 2, 3, 3]. Then the largest element will be [3].</p> <p><b>Sample Input 2:</b> 2 5 5 0 10 1 2 2 6 2 -2 12 -1 1 20 1</p> <p><b>Sample Output 2:</b> 0 1 2 2 10 12 20</p> <p><b>Constraints:</b>  <math>1 \leq T \leq 100</math>  <math>1 \leq N \leq 10^4</math>  <math>1 \leq K \leq N</math>  <math>-10^9 \leq \text{ARR}[i] \leq 10^9</math> </p>	
<b>6</b>	<p><b>Count All Subarrays With Given Sum</b></p> <p><b>Problem Statement:</b> You are given an integer array 'arr' of size 'N' and an integer 'K'. Your task is to find the total number of subarrays of the given array whose sum of elements is equal to k. A subarray is defined as a contiguous block of elements in the array.</p> <p><b>Sample Input 1:</b> 2 4 6 3 1 2 4</p>	Medium

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



	<p>3 3 1 2 3</p> <p><b>Sample output 1:</b></p> <p>2 2</p> <p><b>Explanation:</b></p> <p><b>Test Case 1:</b> Input: 'N' = 4, 'arr' = [3, 1, 2, 4], 'K' = 6 Output: 2 Explanation: The subarrays that sum up to '6' are: [3, 1, 2], and [2, 4].</p> <p><b>Test Case 2:</b> Input: 'N' = 3, 'arr' = [1, 2, 3], 'K' = 3 Output: 2 Explanation: The subarrays that sum up to '7' are: [1, 2], and [3].</p> <p><b>Sample Input 2:</b></p> <p>2 3 7 1 2 3</p> <p>4 9 6 3 5 2</p> <p><b>Sample output 2:</b></p> <p>0 1</p>	
<b>7</b>	<p><b>Find Four Elements That Sums To A Given Value</b></p> <p><b>Problem Statement:</b> You are given an array/list 'ARR' of 'N' integers and an integer value 'TARGET'. You need to check whether there exist four numbers (ARR[i], ARR[j], ARR[k], ARR[l]) such that (0 ≤ i &lt; j &lt; k &lt; l &lt; N) and ARR[i] + ARR[j] + ARR[k] + ARR[l] = 'TARGET'.</p> <p><b>Note:</b></p>	Medium

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



	<p>1. All four numbers should exist at different indices in the given array. 2. The answer is case-sensitive.</p> <p><b>Sample Input 1:</b></p> <p>2 5 9 1 3 3 10 2 6 20 2 4 6 3 1 1</p> <p><b>Sample Output 1:</b></p> <p>Yes No</p> <p><b>Explanation For Sample Input 1:</b></p> <p><b>Test case 1:</b> The elements at indices (0, 1, 2, 4) gives sum 9 i.e, <math>ARR[0] + ARR[1] + ARR[2] + ARR[4] = 9</math>. Hence the answer is Yes.</p> <p><b>Test case 2:</b> None of the combinations of 4 numbers gives 20 as 'TARGET'. Hence the answer is No.</p> <p><b>Sample Input 2:</b></p> <p>2 5 15 0 10 1 2 2 6 20 -2 12 -1 1 20 1</p> <p><b>Sample Output 2:</b></p> <p>Yes Yes</p> <p><b>Constraints:</b></p> <p><math>1 \leq T \leq 10^2</math>  <math>4 \leq N \leq 2 \cdot 10^2</math>  <math>-10^9 \leq ARR[i] \leq 10^9</math>  <math>-10^9 \leq TARGET \leq 10^9</math></p>	
<b>8</b>	<b>Number of Distinct Islands</b>	Medium

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





**Problem Statement:** You are given an  $m \times n$  binary matrix grid. An island is a group of 1's (representing land) connected 4-directionally (horizontal or vertical.) You may assume all four edges of the grid are surrounded by water.

An island is considered to be the same as another if and only if one island can be translated (and not rotated or reflected) to equal the other.

Return the number of distinct islands.

**Example 1:**

1	1	0	0	0
1	1	0	0	0
0	0	0	1	1
0	0	0	1	1

**Input:** grid = [[1,1,0,0,0],[1,1,0,0,0],[0,0,0,1,1],[0,0,0,1,1]]

**Output:** 1

**Example 2:**

1	1	0	1	1
1	0	0	0	0
0	0	0	0	1
1	1	0	1	1

**Input:** grid = [[1,1,0,1,1],[1,0,0,0,0],[0,0,0,0,1],[1,1,0,1,1]]

**Output:** 3

**Constraints:**

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



	<pre>m == grid.length n == grid[i].length 1 &lt;= m, n &lt;= 50 grid[i][j] is either 0 or 1.</pre>																																																																																		
9	<p><b>Valid Sudoku</b></p> <p><b>Problem Statement:</b> Determine if a 9 x 9 Sudoku board is valid. Only the filled cells need to be validated according to the following rules: Each row must contain the digits 1-9 without repetition. Each column must contain the digits 1-9 without repetition. Each of the nine 3 x 3 sub-boxes of the grid must contain the digits 1-9 without repetition.</p> <p><b>Note:</b> A Sudoku board (partially filled) could be valid but is not necessarily solvable. Only the filled cells need to be validated according to the mentioned rules.</p> <p><b>Example 1:</b></p> <table border="1"><tr><td>5</td><td>3</td><td></td><td></td><td>7</td><td></td><td></td><td></td><td></td></tr><tr><td>6</td><td></td><td></td><td>1</td><td>9</td><td>5</td><td></td><td></td><td></td></tr><tr><td></td><td>9</td><td>8</td><td></td><td></td><td></td><td></td><td>6</td><td></td></tr><tr><td>8</td><td></td><td></td><td></td><td>6</td><td></td><td></td><td></td><td>3</td></tr><tr><td>4</td><td></td><td></td><td>8</td><td></td><td>3</td><td></td><td></td><td>1</td></tr><tr><td>7</td><td></td><td></td><td></td><td>2</td><td></td><td></td><td></td><td>6</td></tr><tr><td></td><td>6</td><td></td><td></td><td></td><td></td><td>2</td><td>8</td><td></td></tr><tr><td></td><td></td><td></td><td>4</td><td>1</td><td>9</td><td></td><td></td><td>5</td></tr><tr><td></td><td></td><td></td><td></td><td>8</td><td></td><td></td><td>7</td><td>9</td></tr></table> <p>Input: board =</p> <pre>[["5","3",".",".","7",".",".",".","."], ["6",".",".","1","9","5",".",".","."], [[".","9","8",".",".",".",".","6","."], ["8",".",".","6",".",".","","3"], ["4",".",".","8",".","3",".",".","1"], ["7",".",".","2",".",".","","6"], ["","6","","","","2","8",""], ["","","4","1","9","","","5"], ["","","","8","","","7","9"]]</pre>	5	3			7					6			1	9	5					9	8					6		8				6				3	4			8		3			1	7				2				6		6					2	8					4	1	9			5					8			7	9	Medium
5	3			7																																																																															
6			1	9	5																																																																														
	9	8					6																																																																												
8				6				3																																																																											
4			8		3			1																																																																											
7				2				6																																																																											
	6					2	8																																																																												
			4	1	9			5																																																																											
				8			7	9																																																																											

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



	<pre>,["7",".",".",".", "2",".",".",".", "6"] ,[".","6",".",".",".", "2","8","."] ,[".",".",".", "4","1","9",".",".", "5"] ,[".",".",".", "8",".",".", "7","9"]]</pre> <p>Output: true</p> <p><b>Example 2:</b></p> <p>Input: board =</p> <pre>[["8","3",".",".", "7",".",".",".", ""] ,["6",".",".", "1","9","5",".",".", ""] ,[".", "9","8",".",".", "6","."] ,["8",".",".", "6",".",".", "3"] ,["4",".", "8",".", "3",".", "1"] ,["7",".",".", "2",".",".", "6"] ,[".", "6",".",".", "2","8","."] ,[".", "4","1","9",".", "5"] ,[".", "8",".", "7","9"]]</pre> <p>Output: false</p> <p>Explanation: Same as Example 1, except with the 5 in the top left corner being modified to 8. Since there are two 8's in the top left 3x3 sub-box, it is invalid.</p> <p><b>Constraints:</b></p> <pre>board.length == 9 board[i].length == 9 board[i][j] is a digit 1-9 or '.'.</pre>	
10	<p><b>Insert Delete Get Random O(1)</b></p> <p><b>Problem Statement:</b> Implement the RandomizedSet class:</p> <ul style="list-style-type: none"> <li>RandomizedSet() Initializes the RandomizedSet object.</li> </ul>	Medium

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



- `bool insert(int val)` Inserts an item `val` into the set if not present. Returns `true` if the item was not present, `false` otherwise.
- `bool remove(int val)` Removes an item `val` from the set if present. Returns `true` if the item was present, `false` otherwise.
- `int getRandom()` Returns a random element from the current set of elements (it's guaranteed that at least one element exists when this method is called). Each element must have the same probability of being returned. You must implement the functions of the class such that each function works in average  $O(1)$  time complexity.

**Example 1:****Input**

```
["RandomizedSet", "insert", "remove", "insert", "getRandom",  
"remove", "insert", "getRandom"]  
[[], [1], [2], [2], [], [1], [2], []]
```

**Output**

```
[null, true, false, true, 2, true, false, 2]
```

**Explanation**

```
RandomizedSet randomizedSet = new RandomizedSet();  
randomizedSet.insert(1); // Inserts 1 to the set. Returns true as 1 was  
inserted successfully.  
randomizedSet.remove(2); // Returns false as 2 does not exist in the  
set.  
randomizedSet.insert(2); // Inserts 2 to the set, returns true. Set now  
contains [1,2].  
randomizedSet.getRandom(); // getRandom() should return either 1  
or 2 randomly.  
randomizedSet.remove(1); // Removes 1 from the set, returns true.  
Set now contains [2].  
randomizedSet.insert(2); // 2 was already in the set, so return false.  
randomizedSet.getRandom(); // Since 2 is the only number in the set,  
getRandom() will always return 2.
```

**Constraints:**
$$-2^{31} \leq \text{val} \leq 2^{31} - 1$$

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



	<p>At most <math>2 * 10^5</math> calls will be made to insert, remove, and getRandom. There will be at least one element in the data structure when getRandom is called.</p>	
<b>11</b>	<p><b>Find Duplicates Using Bit Array</b></p> <p><b>Problem Statement:</b> You are given an array/list 'ARR' of N elements where <math>N \leq 32000</math>. 'ARR' can have duplicate entries. Your task is to find duplicate elements, provided you can use a maximum of 4 KiloBytes of extra memory.</p> <p><b>Sample input 1:</b></p> <pre>2 5 4 3 4 2 1 6 1 1 8 6 5 5</pre> <p><b>Sample output 1:</b></p> <pre>4 1 5</pre> <p><b>Explanation For Sample input 1:</b></p> <p><b>For the first test case :</b> There is only one element with duplicate entries, that is 4 on indexes 1 and 3.</p> <p><b>For the second test case :</b> There are two elements with duplicate entries Element 1 exists on indexes 0 and 1. Element 5 exists on indexes 4 and 5.</p> <p><b>Sample input 2:</b></p> <pre>2 5 7 7 4 3 4 6 1 2 3 4 5 7</pre> <p><b>Sample output 2:</b></p> <pre>4 7</pre> <p><b>Constraints</b></p>	Medium

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



	$1 \leq T \leq 100$ $1 \leq N \leq 32000$ $0 \leq \text{'ARR'[i]} \leq 32000$	
<b>12</b>	<p><b>Circular Permutation In Binary Representation</b></p> <p><b>Problem Statement:</b> You are given the two integers 'N' and 'X'. Your task is to find an array 'P' such that:          'P' is a permutation of <math>(0, 1, 2, \dots, 2^N - 1)</math>.          The first element of the array 'P' is 'X', i.e., <math>P[0] = X</math>.          Adjacent elements of 'P' (i.e., <math>P[i]</math> and <math>P[i + 1]</math>) differ by only 1 bit in their binary representation.          The first and the last element (i.e., <math>P[0]</math> and <math>P[2^N - 1]</math>) are also considered as adjacent elements.</p> <p><b>Note:</b>          For <math>N = 2</math>, <math>[0, 1, 2, 3]</math>, <math>[0, 2, 3, 1]</math>, <math>[1, 2, 3, 0]</math> are some of the valid permutations but <math>[0, 0, 1, 2]</math>, <math>[1, 2, 3, 4]</math>, <math>[1, 1, 1, 3]</math> are not.          It is guaranteed that an array 'P' always exists with the given requirements.</p> <p><b>Sample Input 1:</b>          1          1 1</p> <p><b>Sample Output 1:</b>          1 0</p> <p><b>Explanation For Sample Output 1:</b>          The binary representation of the array <math>[1, 0]</math> is also <math>(1, 0)</math> in which adjacent elements differ by only 1 bit.</p> <p><b>Sample Input 2:</b>          1          2 0</p> <p><b>Sample Output 2:</b>          0 1 3 2</p> <p><b>Constraints:</b>  <math>1 \leq T \leq 100</math>  <math>1 \leq N \leq 13</math>  <math>0 \leq X &lt; 2^N</math></p>	Medium

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



13	<p><b>Best Time to Buy and Sell Stock II</b></p> <p><b>Problem Statement:</b> You are given an integer array prices where prices[i] is the price of a given stock on the ith day.</p> <p>On each day, you may decide to buy and/or sell the stock. You can only hold at most one share of the stock at any time. However, you can buy it then immediately sell it on the same day.</p> <p>Find and return the maximum profit you can achieve.</p> <p><b>Example 1:</b></p> <p>Input: prices = [7,1,5,3,6,4]</p> <p>Output: 7</p> <p>Explanation: Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit = 5-1 = 4.</p> <p>Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3.</p> <p>Total profit is 4 + 3 = 7.</p> <p><b>Example 2:</b></p> <p>Input: prices = [1,2,3,4,5]</p> <p>Output: 4</p> <p>Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4.</p> <p>Total profit is 4.</p> <p><b>Example 3:</b></p> <p>Input: prices = [7,6,4,3,1]</p> <p>Output: 0</p> <p>Explanation: There is no way to make a positive profit, so we never buy the stock to achieve the maximum profit of 0.</p> <p><b>Constraints:</b></p>	Medium

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



	$1 \leq \text{prices.length} \leq 3 * 10^4$ $0 \leq \text{prices}[i] \leq 10^4$	
<b>14</b>	<p><b>Process Tasks Using Servers</b></p> <p><b>Problem Statement:</b> You are given two 0-indexed integer arrays <code>servers</code> and <code>tasks</code> of lengths <code>n</code> and <code>m</code> respectively. <code>servers[i]</code> is the weight of the <code>i</code>th server, and <code>tasks[j]</code> is the time needed to process the <code>j</code>th task in seconds. Tasks are assigned to the servers using a task queue. Initially, all servers are free, and the queue is empty. At second <code>j</code>, the <code>j</code>th task is inserted into the queue (starting with the 0th task being inserted at second 0). As long as there are free servers and the queue is not empty, the task in the front of the queue will be assigned to a free server with the smallest weight, and in case of a tie, it is assigned to a free server with the smallest index.</p> <p>If there are no free servers and the queue is not empty, we wait until a server becomes free and immediately assign the next task. If multiple servers become free at the same time, then multiple tasks from the queue will be assigned in order of insertion following the weight and index priorities above. A server that is assigned task <code>j</code> at second <code>t</code> will be free again at second <code>t + tasks[j]</code>. Build an array <code>ans</code> of length <code>m</code>, where <code>ans[j]</code> is the index of the server the <code>j</code>th task will be assigned to. Return the array <code>ans</code>.</p> <p><b>Example 1:</b></p> <p>Input: <code>servers = [3,3,2]</code>, <code>tasks = [1,2,3,2,1,2]</code>  Output: <code>[2,2,0,2,1,2]</code>  Explanation: Events in chronological order go as follows:</p> <ul style="list-style-type: none"> <li>- At second 0, task 0 is added and processed using server 2 until second 1.</li> <li>- At second 1, server 2 becomes free. Task 1 is added and processed using server 2 until second 3.</li> <li>- At second 2, task 2 is added and processed using server 0 until second 5.</li> </ul>	Medium

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





	<ul style="list-style-type: none"> <li>- At second 3, server 2 becomes free. Task 3 is added and processed using server 2 until second 5.</li> <li>- At second 4, task 4 is added and processed using server 1 until second 5.</li> <li>- At second 5, all servers become free. Task 5 is added and processed using server 2 until second 7.</li> </ul> <p><b>Example 2:</b></p> <p>Input: servers = [5,1,4,3,2], tasks = [2,1,2,4,5,2,1]</p> <p>Output: [1,4,1,4,1,3,2]</p> <p>Explanation: Events in chronological order go as follows:</p> <ul style="list-style-type: none"> <li>- At second 0, task 0 is added and processed using server 1 until second 2.</li> <li>- At second 1, task 1 is added and processed using server 4 until second 2.</li> <li>- At second 2, servers 1 and 4 become free. Task 2 is added and processed using server 1 until second 4.</li> <li>- At second 3, task 3 is added and processed using server 4 until second 7.</li> <li>- At second 4, server 1 becomes free. Task 4 is added and processed using server 1 until second 9.</li> <li>- At second 5, task 5 is added and processed using server 3 until second 7.</li> <li>- At second 6, task 6 is added and processed using server 2 until second 7.</li> </ul> <p><b>Constraints:</b></p> <p>servers.length == n  tasks.length == m  <math>1 \leq n, m \leq 2 * 10^5</math>  <math>1 \leq \text{servers}[i], \text{tasks}[j] \leq 2 * 10^5</math></p>	
15	<b>Seat Reservation Manager</b>	Medium

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



**Problem Statement:** Design a system that manages the reservation state of  $n$  seats that are numbered from 1 to  $n$ .  
Implement the SeatManager class:  
SeatManager(int n) Initializes a SeatManager object that will manage  $n$  seats numbered from 1 to  $n$ . All seats are initially available.  
int reserve() Fetches the smallest-numbered unreserved seat, reserves it, and returns its number.  
void unreserve(int seatNumber) Unreserves the seat with the given seatNumber.

#### Example 1:

Input

```
["SeatManager", "reserve", "reserve", "unreserve", "reserve",  
"reserve", "reserve", "reserve", "unreserve"]  
[[5], [], [], [2], [], [], [], [], [5]]
```

#### Output

```
[null, 1, 2, null, 2, 3, 4, 5, null]
```

#### Explanation

SeatManager seatManager = new SeatManager(5); // Initializes a SeatManager with 5 seats.

seatManager.reserve(); // All seats are available, so return the lowest numbered seat, which is 1.

seatManager.reserve(); // The available seats are [2,3,4,5], so return the lowest of them, which is 2.

seatManager.unreserve(2); // Unreserve seat 2, so now the available seats are [2,3,4,5].

seatManager.reserve(); // The available seats are [2,3,4,5], so return the lowest of them, which is 2.

seatManager.reserve(); // The available seats are [3,4,5], so return the lowest of them, which is 3.

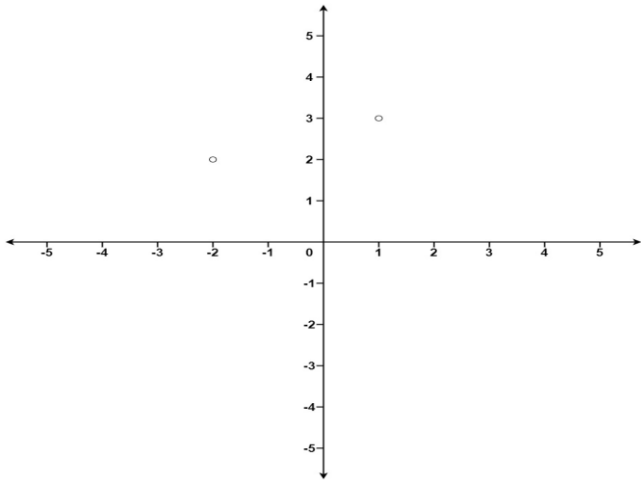
seatManager.reserve(); // The available seats are [4,5], so return the lowest of them, which is 4.

seatManager.reserve(); // The only available seat is seat 5, so return 5.

seatManager.unreserve(5); // Unreserve seat 5, so now the available seats are [5].

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



	<p><b>Constraints:</b></p> <p><math>1 \leq n \leq 10^5</math></p> <p><math>1 \leq \text{seatNumber} \leq n</math></p> <p>For each call to reserve, it is guaranteed that there will be at least one unreserved seat.</p> <p>For each call to unreserve, it is guaranteed that seatNumber will be reserved.</p> <p>At most 105 calls in total will be made to reserve and unreserve.</p>	
<b>16</b>	<p><b>K Closest Points to Origin</b></p> <p><b>Problem Statement:</b> Given an array of points where <math>\text{points}[i] = [x_i, y_i]</math> represents a point on the X-Y plane and an integer <math>k</math>, return the <math>k</math> closest points to the origin <math>(0, 0)</math>.</p> <p>The distance between two points on the X-Y plane is the Euclidean distance (i.e., <math>\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}</math>).</p> <p>You may return the answer in any order. The answer is guaranteed to be unique (except for the order that it is in).</p> <p><b>Example 1:</b></p>  <p>Input: <math>\text{points} = [[1, 3], [-2, 2]]</math>, <math>k = 1</math></p> <p>Output: <math>[[-2, 2]]</math></p>	Medium

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



	<p>Explanation:</p> <p>The distance between (1, 3) and the origin is <math>\sqrt{10}</math>.</p> <p>The distance between (-2, 2) and the origin is <math>\sqrt{8}</math>.</p> <p>Since <math>\sqrt{8} &lt; \sqrt{10}</math>, (-2, 2) is closer to the origin.</p> <p>We only want the closest <math>k = 1</math> points from the origin, so the answer is just <math>[[-2, 2]]</math>.</p> <p><b>Example 2:</b></p> <p>Input: points = <math>[[3, 3], [5, -1], [-2, 4]]</math>, <math>k = 2</math></p> <p>Output: <math>[[3, 3], [-2, 4]]</math></p> <p>Explanation: The answer <math>[-2, 4], [3, 3]</math> would also be accepted.</p> <p><b>Constraints:</b></p> <p><math>1 \leq k \leq \text{points.length} \leq 10^4</math></p> <p><math>-10^4 \leq x_i, y_i \leq 10^4</math></p>	
<b>17</b>	<p><b>Exam Room</b></p> <p>Problem Statement: There is an exam room with <math>n</math> seats in a single row labeled from 0 to <math>n - 1</math>. When a student enters the room, they must sit in the seat that maximizes the distance to the closest person. If there are multiple such seats, they sit in the seat with the lowest number. If no one is in the room, then the student sits at seat number 0. Design a class that simulates the mentioned exam room.</p> <p>Implement the ExamRoom class:</p> <ul style="list-style-type: none"> <li>• ExamRoom(int n) Initializes the object of the exam room with the number of the seats <math>n</math>.</li> <li>• int seat() Returns the label of the seat at which the next student will set.</li> <li>• void leave(int p) Indicates that the student sitting at seat <math>p</math> will leave the room. It is guaranteed that there will be a student sitting at seat <math>p</math>.</li> </ul> <p><b>Example 1:</b></p> <p>Input</p> <p><math>["\text{ExamRoom}", "\text{seat}", "\text{seat}", "\text{seat}", "\text{seat}", "\text{leave}", "\text{seat}"]</math></p>	Medium

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



	<pre>[[10], [], [], [], [], [4], []]</pre> <p><b>Output</b></p> <pre>[null, 0, 9, 4, 2, null, 5]</pre> <p><b>Explanation</b></p> <pre>ExamRoom examRoom = new ExamRoom(10); examRoom.seat(); // return 0, no one is in the room, then the student sits at seat number 0. examRoom.seat(); // return 9, the student sits at the last seat number 9. examRoom.seat(); // return 4, the student sits at the last seat number 4. examRoom.seat(); // return 2, the student sits at the last seat number 2. examRoom.leave(4); examRoom.seat(); // return 5, the student sits at the last seat number 5.</pre> <p><b>Constraints:</b></p> <p>1 &lt;= n &lt;= 109</p> <p>It is guaranteed that there is a student sitting at seat p.</p> <p>At most 104 calls will be made to seat and leave.</p>	
<b>18</b>	<p><b>The Celebrity Problem</b></p> <p><b>Problem statement</b></p> <p>There are 'N' people at a party. Each person has been assigned a unique id between 0 to 'N' - 1(both inclusive). A celebrity is a person who is known to everyone but does not know anyone at the party. Given a helper function 'knows(A, B)', It will returns "true" if the person having id 'A' know the person having id 'B' in the party, "false" otherwise. Your task is to find out the celebrity at the party. Print the id of the celebrity, if there is no celebrity at the party then print -1.</p> <p><b>Note:</b></p> <p>1. The helper function 'knows' is already implemented for you.</p>	Medium

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



2. 'knows(A, B)' returns "false", if A doesn't know B.
3. You should not implement helper function 'knows', or speculate about its implementation.
4. You should minimize the number of calls to function 'knows(A, B)'.
5. There are at least 2 people at the party.
6. At most one celebrity will exist.

**Sample Input 1:**

1  
2

Call function 'knows(0, 1)' // returns false

Call function 'knows(1, 0)' // returns true

**Sample Output 1:**

0

**Explanation For Sample Input 1:**

In the first test case, there are 2 people at the party. When we call function knows(0,1), it returns false. That means the person having id '0' does not know a person having id '1'. Similarly, the person having id '1' knows a person having id '0' as knows(1,0) returns true. Thus a person having id '0' is a celebrity because he is known to everyone at the party but doesn't know anyone.

**Sample Input 2:**

1  
2

Call 'knows(0, 1)' // returns true

Call 'knows(1, 0)' // returns true

2

Call 'knows(0, 1)' // returns false

Call 'knows(1, 0)' // returns false

**Sample Output 2:**

-1  
-1

**Explanation For Sample Input 2:**

In first test case, there are 2 people at the party. The person having id '0' knows a person having id '1'. The person having id '1' knows a person having id '0'. Thus there is no celebrity at the party, because both know each other.

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



	<p>In second test case, there are 2 people at the party. The person having id '0' does not know a person having id '1'. The person having id '1' also does not know a person having id '0'. Thus there is no celebrity at the party, because both does not know each other.</p> <p><b>Constraints:</b></p> <p><math>1 \leq T \leq 50</math></p> <p><math>2 \leq N \leq 10^4</math></p>	
<b>19</b>	<p><b>Group Anagrams</b></p> <p><b>Problem Statement:</b> Given an array of strings <code>strs</code>, group the anagrams together. You can return the answer in any order.</p> <p>An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.</p> <p><b>Example 1:</b></p> <p>Input: <code>strs = ["eat","tea","tan","ate","nat","bat"]</code></p> <p>Output: <code>[["bat"],["nat","tan"],["ate","eat","tea"]]</code></p> <p><b>Example 2:</b></p> <p>Input: <code>strs = [""]</code></p> <p>Output: <code>[[""]]</code></p> <p><b>Constraints:</b></p> <p><math>1 \leq \text{strs.length} \leq 104</math></p> <p><math>0 \leq \text{strs}[i].\text{length} \leq 100</math></p> <p><code>strs[i]</code> consists of lowercase English letters.</p>	Medium
<b>20</b>	<p><b>Search Pattern</b></p> <p><b>Problem Statement:</b> You're given two strings, '<b>text</b>' of length '<b>n</b>' and '<b>pattern</b>' of length '<b>m</b>', consisting of lowercase characters.</p> <p>Find all the occurrences of the string 'pattern' in 'text'.</p>	Medium

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



	<p>For each occurrence, print the index from where it starts in the string 'text' (1 - indexed).</p> <p><b>Sample Input 1:</b> cxyzghxyzvjxyz xyz</p> <p><b>Sample Output 1:</b> 3 2 7 13</p> <p><b>Explanation Of Sample Input 1 :</b> The pattern 'pattern' = "xyz" appears at 3 positions in 'text'.</p> <p><b>Sample Input 2 :</b> ababacabab aba</p> <p><b>Sample Output 2 :</b> 3 1 3 7</p> <p><b>Explanation Of Sample Input 2 :</b> Here we have an overlap between the first occurrence (at position 1) and the second occurrence (at position 3), and we are considering both.</p> <p><b>Sample Input 3 :</b> abcd xy</p> <p><b>Sample Output 3 :</b> 0</p> <p><b>Expected time complexity:</b> The expected time complexity is <math>O('n' + 'm')</math>.</p>	
--	--	--

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





	<b>Constraints:</b> $1 \leq 'n' \leq 10^5$ $1 \leq 'm' \leq 'n'$	
<b>21</b>	<b>Count With K Different Characters</b>  <p><b>Problem Statement:</b> You are given a string '<i>str</i>' of lowercase alphabets and an integer '<i>k</i>' .  Your task is to return the count all the possible substrings that have exactly 'k' distinct characters.</p> <p><b>Sample Input 1 :</b>  aacfssa  3</p> <p><b>Sample Output 1 :</b>  5</p> <p><b>Explanation of The Sample Input 1:</b>  Given 'str' = "aacfssa". We can see that the substrings with only 3 distinct characters are {aacf, acf, cfs, cfss, fssa}.</p> <p>Therefore, the answer will be 5.</p> <p><b>Sample Input 2 :</b>  qffds  4</p> <p><b>Sample Output 2 :</b>  1</p> <p><b>Constraints:</b>  <math>1 \leq  str  \leq 10^5</math>  <math>1 \leq k \leq 26</math></p>	Medium
<b>22</b>	<b>Count Distinct Substrings</b>  <p><b>Problem Statement:</b> Given a string 'S', you are supposed to return the number of distinct substrings(including empty substring) of the given string.</p> <p><b>Sample Input 1 :</b>  2  sds</p>	Medium

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



	<p>abc</p> <p><b>Sample Output 1 :</b></p> <p>6</p> <p>7</p> <p><b>Explanation of Sample Input 1 :</b></p> <p>In the first test case, the 6 distinct substrings are { 's', 'd', "sd", "ds", "sds", "" }</p> <p>In the second test case, the 7 distinct substrings are { 'a', 'b', 'c', "ab", "bc", "abc", "" }.</p> <p><b>Sample Input 2 :</b></p> <p>2</p> <p>aa</p> <p>abab</p> <p><b>Sample Output 2 :</b></p> <p>3</p> <p>8</p> <p><b>Explanation of Sample Input 2 :</b></p> <p>In the first test case, the two distinct substrings are { 'a', "aa", "" }.</p> <p>In the second test case, the seven distinct substrings are { 'a', 'b', "ab", "ba", "aba", "bab", "abab", "" }</p> <p><b>Constraints :</b></p> <p><math>1 \leq T \leq 5</math></p> <p><math>\leq  S  \leq 10^3</math></p>	
<b>23</b>	<p><b>Longest Substring Without Repeating Characters</b></p> <p><b>Problem Statement:</b> Given a string s, find the length of the longest substring without repeating characters.</p> <p><b>Example 1:</b></p> <p>Input: s = "abcabcbb"</p> <p>Output: 3</p> <p>Explanation: The answer is "abc", with the length of 3.</p> <p><b>Example 2:</b></p> <p>Input: s = "bbbbbb"</p> <p>Output: 1</p>	Medium

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



	<p>Explanation: The answer is "b", with the length of 1.</p> <p><b>Constraints:</b></p> <p><math>0 \leq s.length \leq 5 * 10^4</math></p> <p>s consists of English letters, digits, symbols and spaces.</p>	
<b>24</b>	<p><b>Flatten 2D Vector</b></p> <p><b>Problem Statement:</b> Design an iterator to flatten a 2D vector. It should support the next and hasNext operations.</p> <p>Implement the Vector2D class:</p> <ul style="list-style-type: none"><li>• Vector2D(int[][] vec) initializes the object with the 2D vector vec.</li><li>• next() returns the next element from the 2D vector and moves the pointer one step forward. You may assume that all the calls to next are valid.</li><li>• hasNext() returns true if there are still some elements in the vector, and false otherwise.</li></ul> <p><b>Example 1:</b></p> <p>Input</p> <p>["Vector2D", "next", "next", "next", "hasNext", "hasNext", "next", "hasNext"]</p> <p>[[[1, 2], [3], [4]], [], [], [], [], [], [], []]</p> <p>Output</p> <p>[null, 1, 2, 3, true, true, 4, false]</p> <p>Explanation</p>	Medium



	<pre>Vector2D vector2D = new Vector2D([[1, 2], [3], [4]]);  vector2D.next(); // return 1  vector2D.next(); // return 2  vector2D.next(); // return 3  vector2D.hasNext(); // return True  vector2D.hasNext(); // return True  vector2D.next(); // return 4  vector2D.hasNext(); // return False</pre> <p><b>Constraints:</b></p> <p><math>0 \leq \text{vec.length} \leq 200</math></p> <p><math>0 \leq \text{vec}[i].\text{length} \leq 500</math></p> <p><math>-500 \leq \text{vec}[i][j] \leq 500</math></p> <p>At most 105 calls will be made to next and hasNext.</p>	
<b>25</b>	<p><b>Sort Characters By Frequency</b></p> <p><b>Problem Statement:</b> Given a string <i>s</i>, sort it in decreasing order based on the frequency of the characters. The frequency of a character is the number of times it appears in the string. Return the sorted string. If there are multiple answers, return any of them.</p> <p><b>Example 1:</b> Input: <i>s</i> = "tree" Output: "eert" Explanation: 'e' appears twice while 'r' and 't' both appear once. So 'e' must appear before both 'r' and 't'. Therefore "eetr" is also a valid answer.</p>	Medium



	<p><b>Example 2:</b></p> <p>Input: s = "cccaaa"</p> <p>Output: "aaaccc"</p> <p>Explanation: Both 'c' and 'a' appear three times, so both "cccaaa" and "aaaccc" are valid answers.</p> <p>Note that "cacaca" is incorrect, as the same characters must be together.</p> <p><b>Constraints:</b></p> <p><math>1 \leq s.length \leq 5 * 10^5</math></p> <p>s consists of uppercase and lowercase English letters and digits.</p>	
26	<p><b>Zuma Game</b></p> <p><b>Problem statement</b></p> <p>You have several balls on the table in the form of a string named 'BOARD'. The colors of the balls can be red(R), blue(B), green(G), white(W), and yellow(Y). You also have several balls in your hand in the form of a string named 'hand'. Each time, you can do the following operations:</p> <ol style="list-style-type: none"><li>1. Choose a ball from the string 'HAND', and insert it anywhere on the string 'BOARD'.</li><li>2. If there is a group of strictly more than 2 balls of the same color touching each other, remove them from the string 'BOARD'. Keep doing this until the string 'board' becomes empty or no more balls can satisfy this condition.</li></ol> <p>Your task is to find the minimum number of insertions required to make the 'BOARD' empty. If it is not possible to make the string 'BOARD' empty, then print -1.</p> <p>Note:</p> <ol style="list-style-type: none"><li>1) Both strings will be non-empty and will only contain the characters 'R', 'B', 'G', 'W', and 'Y'.</li><li>2) Initially, the string 'BOARD' won't have more than 2 balls of the same colors touching each other.</li></ol>	Medium

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



	<p><b>Sample Input 1:</b></p> <p>1 YYRR YR</p> <p><b>Sample Output 1:</b></p> <p>2</p> <p><b>Explanation for sample input 1:</b></p> <p>One of the possible order of insertions: YYRR -&gt; YY[Y]RR -&gt; RR -&gt; RR[R] -&gt; empty. You required 2 insertions. There is no possible solution with only one insertion.</p> <p><b>Sample Input 2:</b></p> <p>1 RR YG</p> <p><b>Sample Output 2:</b></p> <p>-1</p> <p><b>Explanation for sample input 2:</b></p> <p>It is not possible to make the string board empty.</p> <p><b>Constraints:</b></p> <p>1 &lt;= T &lt;= 10 1 &lt;=  BOARD  &lt;= 10 1 &lt;=  HAND  &lt;= 4</p>	
27	<p><b>Replace Question Marks in String to Minimize Its Value</b></p> <p><b>Problem Statement:</b> You are given a string s. s[i] is either a lowercase English letter or '?'. For a string t having length m containing only lowercase English letters, we define the function cost(i) for an index i as the number of characters equal to t[i] that appeared before it, i.e. in the range [0, i - 1]. The value of t is the sum of cost(i) for all indices i.</p> <p><b>Example 1:</b></p> <p><b>Input:</b> s = "???"</p> <p><b>Output:</b> "abc"</p> <p><b>Explanation:</b> In this example, we can replace the occurrences of '?' to make s equal to "abc".</p>	Medium

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



	<p>For "abc", <math>\text{cost}(0) = 0</math>, <math>\text{cost}(1) = 0</math>, and <math>\text{cost}(2) = 0</math>.</p> <p>The value of "abc" is 0.</p> <p>Some other modifications of s that have a value of 0 are "cba", "abz", and, "hey".</p> <p>Among all of them, we choose the lexicographically smallest.</p> <p><b>Example 2:</b></p> <p><b>Input:</b> s = "a?a?"</p> <p><b>Output:</b> "abac"</p> <p><b>Explanation:</b> In this example, the occurrences of '?' can be replaced to make s equal to "abac".</p> <p>For "abac", <math>\text{cost}(0) = 0</math>, <math>\text{cost}(1) = 0</math>, <math>\text{cost}(2) = 1</math>, and <math>\text{cost}(3) = 0</math>.</p> <p>The value of "abac" is 1.</p> <p><b>Constraints:</b></p> <p><math>1 \leq \text{s.length} \leq 105</math></p> <p>s[i] is either a lowercase English letter or '?'.</p>	
28	<p><b>Search Suggestions System</b></p> <p><b>Problem Statement:</b> You are given an array of strings products and a string searchWord.</p> <p>Design a system that suggests at most three product names from products after each character of searchWord is typed.</p> <p>Suggested products should have common prefix with searchWord.</p> <p>If there are more than three products with a common prefix return the three lexicographically minimums products.</p> <p>Return a list of lists of the suggested products after each character of searchWord is typed.</p> <p><b>Example 1:</b></p> <p><b>Input:</b> products = ["mobile", "mouse", "moneypot", "monitor", "mousepad"], searchWord = "mouse"</p> <p><b>Output:</b> [ ["mobile", "moneypot", "monitor"], ["mobile", "moneypot", "monitor"],</p>	Medium

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



	<p><code>["mouse","mousepad"],["mouse","mousepad"],["mouse","mousepad"]]</code></p> <p><b>Explanation:</b> products sorted lexicographically = <code>["mobile","moneypot","monitor","mouse","mousepad"]</code>.          After typing m and mo all products match and we show user <code>["mobile","moneypot","monitor"]</code>.          After typing mou, mous and mouse the system suggests <code>["mouse","mousepad"]</code>.</p> <p><b>Example 2:</b>  <b>Input:</b> products = <code>["havana"]</code>, searchWord = <code>"havana"</code>  <b>Output:</b>  <code>["havana"],["havana"],["havana"],["havana"],["havana"],["havana"]]</code>  <b>Explanation:</b> The only word "havana" will be always suggested while typing the search word.</p> <p><b>Constraints:</b></p> <p><code>1 &lt;= products.length &lt;= 1000</code>  <code>1 &lt;= products[i].length &lt;= 3000</code>  <code>1 &lt;= sum(products[i].length) &lt;= 2 * 10^4</code>          All the strings of products are unique.  <code>products[i]</code> consists of lowercase English letters.  <code>1 &lt;= searchWord.length &lt;= 1000</code>  <code>searchWord</code> consists of lowercase English letters.</p>	
29	<p><b>Design a Food Rating System</b></p> <p><b>Problem Statement:</b> Design a food rating system that can do the following:</p> <ul style="list-style-type: none"> <li>• Modify the rating of a food item listed in the system.</li> <li>• Return the highest-rated food item for a type of cuisine in the system.</li> </ul> <p>Implement the FoodRatings class:</p> <ul style="list-style-type: none"> <li>• <code>FoodRatings(String[] foods, String[] cuisines, int[] ratings)</code>              Initializes the system. The food items are described by foods, cuisines and ratings, all of which have a length of n.</li> <li>• <code>foods[i]</code> is the name of the ith food,</li> </ul>	Medium

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





- `cuisines[i]` is the type of cuisine of the `i`th food, and
- `ratings[i]` is the initial rating of the `i`th food.
- `void changeRating(String food, int newRating)` Changes the rating of the food item with the name `food`.
- `String highestRated(String cuisine)` Returns the name of the food item that has the highest rating for the given type of cuisine. If there is a tie, return the item with the lexicographically smaller name.

Note that a string `x` is lexicographically smaller than string `y` if `x` comes before `y` in dictionary order, that is, either `x` is a prefix of `y`, or if `i` is the first position such that `x[i] != y[i]`, then `x[i]` comes before `y[i]` in alphabetic order.

**Example 1:**

Input

```
["FoodRatings", "highestRated", "highestRated", "changeRating",  
"highestRated", "changeRating", "highestRated"]  
[[["kimchi", "miso", "sushi", "moussaka", "ramen", "bulgogi"],  
["korean", "japanese", "japanese", "greek", "japanese", "korean"], [9,  
12, 8, 15, 14, 7]], ["korean"], ["japanese"], ["sushi", 16],  
["japanese"], ["ramen", 16], ["japanese"]]
```

**Output**

```
[null, "kimchi", "ramen", null, "sushi", null, "ramen"]
```

**Explanation**

```
FoodRatings foodRatings = new FoodRatings(["kimchi", "miso",  
"sushi", "moussaka", "ramen", "bulgogi"], ["korean", "japanese",  
"japanese", "greek", "japanese", "korean"], [9, 12, 8, 15, 14, 7]);  
foodRatings.highestRated("korean"); // return "kimchi"  
    // "kimchi" is the highest rated korean food  
    with a rating of 9.  
foodRatings.highestRated("japanese"); // return "ramen"  
    // "ramen" is the highest rated japanese  
    food with a rating of 14.  
foodRatings.changeRating("sushi", 16); // "sushi" now has a rating of  
16.  
foodRatings.highestRated("japanese"); // return "sushi"
```

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



	<pre>// "sushi" is the highest rated japanese food with a rating of 16. foodRatings.changeRating("ramen", 16); // "ramen" now has a rating of 16. foodRatings.highestRated("japanese"); // return "ramen" // Both "sushi" and "ramen" have a rating of 16. // However, "ramen" is lexicographically smaller than "sushi".</pre> <p><b>Constraints:</b></p> <p>1 ≤ n ≤ 2 * 10<sup>4</sup>  n == foods.length == cuisines.length == ratings.length  1 ≤ foods[i].length, cuisines[i].length ≤ 10  foods[i], cuisines[i] consist of lowercase English letters.  1 ≤ ratings[i] ≤ 10<sup>8</sup>  All the strings in foods are distinct.</p>	
30	<p><b>Encode and Decode TinyURL</b></p> <p><b>Problem Statement:</b> This is a companion problem to the System Design problem: Design TinyURL. TinyURL is a URL shortening service where you enter a URL such as <a href="https://leetcode.com/problems/design-tinyurl">https://leetcode.com/problems/design-tinyurl</a> and it returns a short URL such as <a href="http://tinyurl.com/4e9iAk">http://tinyurl.com/4e9iAk</a>. Design a class to encode a URL and decode a tiny URL. There is no restriction on how your encode/decode algorithm should work. You just need to ensure that a URL can be encoded to a tiny URL and the tiny URL can be decoded to the original URL.</p> <p><b>Implement the Solution class:</b></p> <ul style="list-style-type: none"> <li>• Solution() Initializes the object of the system.</li> <li>• String encode(String longUrl) Returns a tiny URL for the given longUrl.</li> <li>• String decode(String shortUrl) Returns the original long URL for the given shortUrl. It is guaranteed that the given shortUrl was encoded by the same object.</li> </ul>	Medium

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



	<p><b>Example 1:</b></p> <p><b>Input:</b> url = "https://leetcode.com/problems/design-tinyurl"</p> <p><b>Output:</b> "https://leetcode.com/problems/design-tinyurl"</p> <p><b>Explanation:</b></p> <pre>Solution obj = new Solution(); string tiny = obj.encode(url); // returns the encoded tiny url. string ans = obj.decode(tiny); // returns the original url after decoding it.</pre> <p><b>Constraints:</b></p> <p>1 &lt;= url.length &lt;= 104 url is guaranteed to be a valid URL.</p>	
--	--	--