**SDE ASSIGNMENT 18-11-2024**

**Non Repeating Character**

Difficulty: **Easy**Accuracy: **40.43%**Submissions: **230K+**Points: **2**

Given a string **s** consisting of **lowercase**Latin Letters. Return the first non-repeating character in **s**. If there is no non-repeating character, return **'$'.**  
Note:When you return '$' driver code will output -1.

**Examples:**

**Input:** s = "geeksforgeeks"

**Output:** 'f'

**Explanation:** In the given string, 'f' is the first character in the string which does not repeat.

class Solution {

// Function to find the first non-repeating character in a string.

static char nonRepeatingChar(String s) {

HashMap<Character,Integer> map=new HashMap<>();

char[] a=s.toCharArray();

for(char c:a){

map.put(c,map.getOrDefault(c,0)+1);

}

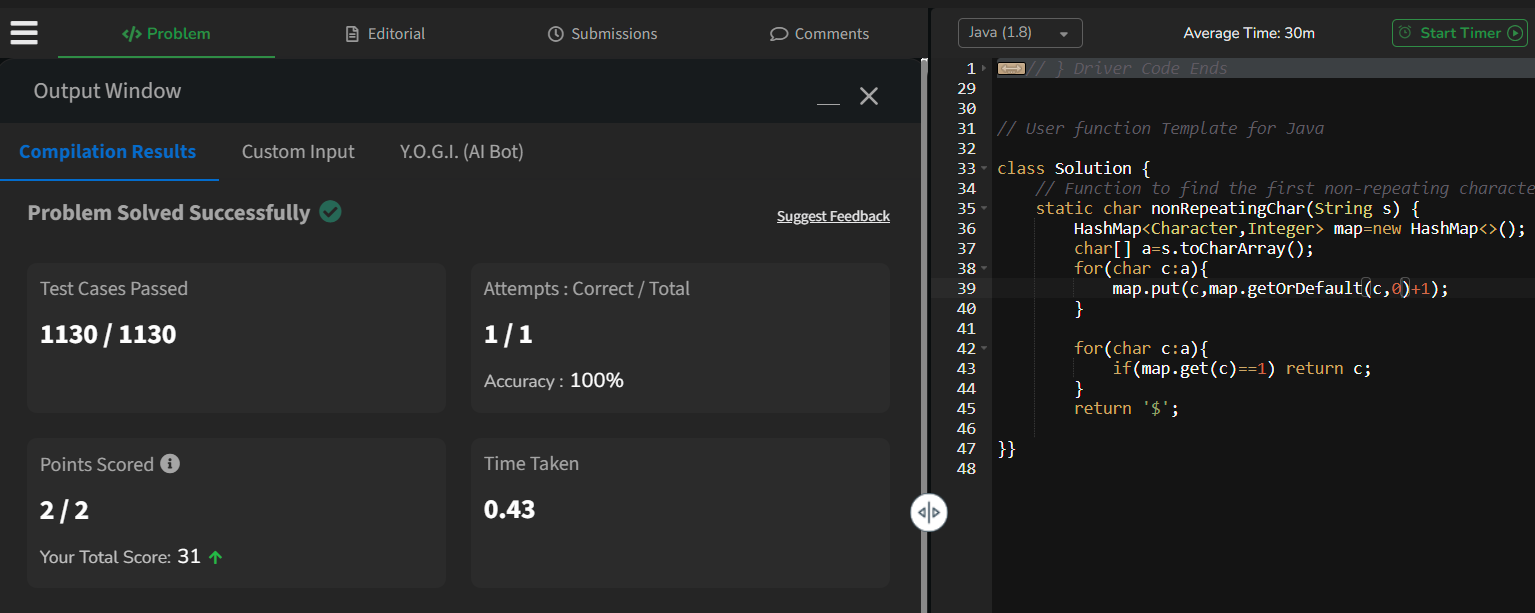
for(char c:a){

if(map.get(c)==1) return c;

}

return '$';

}}



**Bubble Sort**

Difficulty: **Easy**Accuracy: **59.33%**Submissions: **236K+**Points: **2**

Given an array, **arr[]**. Sort the array using bubble sort algorithm.

**Examples :**

**Input**: arr[] = [4, 1, 3, 9, 7]

**Output**: [1, 3, 4, 7, 9]

**Input**: arr[] = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

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

class Solution {

// Function to sort the array using bubble sort algorithm.

public static void bubbleSort(int arr[]) {

for(int i=0;i<arr.length;i++){

for(int j=0;j<arr.length-i-1;j++){

if(arr[j]>arr[j+1]){

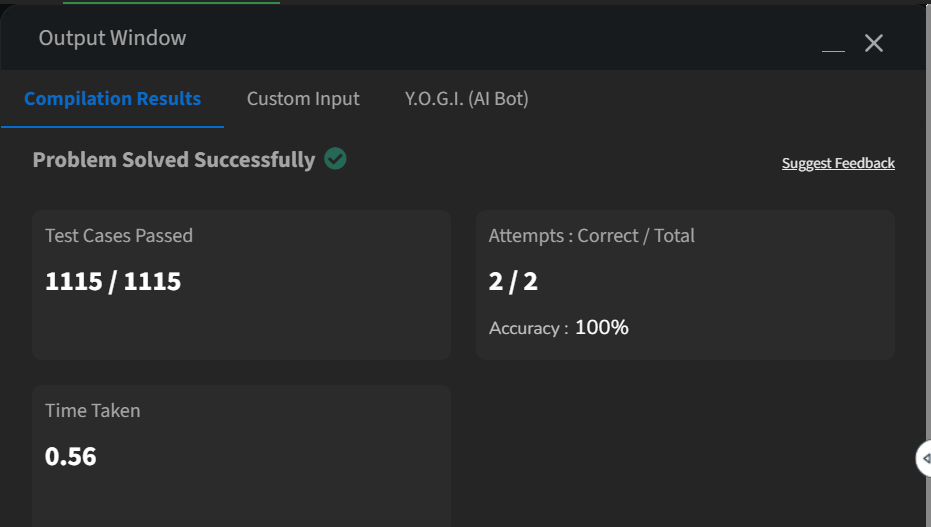
int temp=arr[j];arr[j]=arr[j+1];arr[j+1]=temp;

}

}

}

}

}

**Selection Sort**

Difficulty: **Easy**Accuracy: **64.33%**Submissions: **150K+**Points: **2**

Given an unsorted array of size N, use selection sort to sort arr[] in increasing order.

**Example 1:**

**Input:**

N = 5

arr[] = {4, 1, 3, 9, 7}

**Output:**

1 3 4 7 9

**Explanation:**

Maintain sorted (in bold) and unsorted subarrays.

class Solution

{

int select(int arr[], int i)

{

int min=Integer.MAX\_VALUE,index=0;

for(int j=i;j<arr.length;j++){

if(arr[j]<min){

min=arr[j];index=j;

}

}

return index;

}

void selectionSort(int arr[], int n)

{

for(int i=0;i<n;i++){

int t=arr[i];

int a=select(arr,i);

if(a==i) continue;

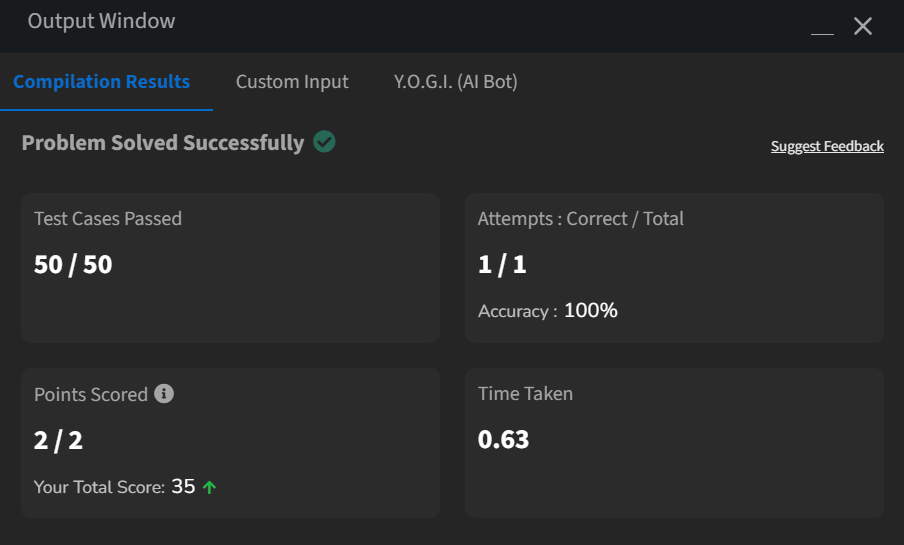
arr[i]=arr[a];

arr[a]=t;

}

}

}



**Kth largest element**

Difficulty: **Medium**Accuracy: **64.92%**Submissions: **22K+**Points: **4**

Given an array **arr**[] and a number **k**. The task is to find the **kth** largest element in the array.

**Examples:**

**Input:** arr[] = [3, 5, 4, 2, 9], k = 3

**Output:** 4

**Explanation:** Third largest element in the array is 4.

class Solution {

public static int KthLargest(int arr[], int k) {

PriorityQueue <Integer> pq=new PriorityQueue<>();

for(int i:arr) {

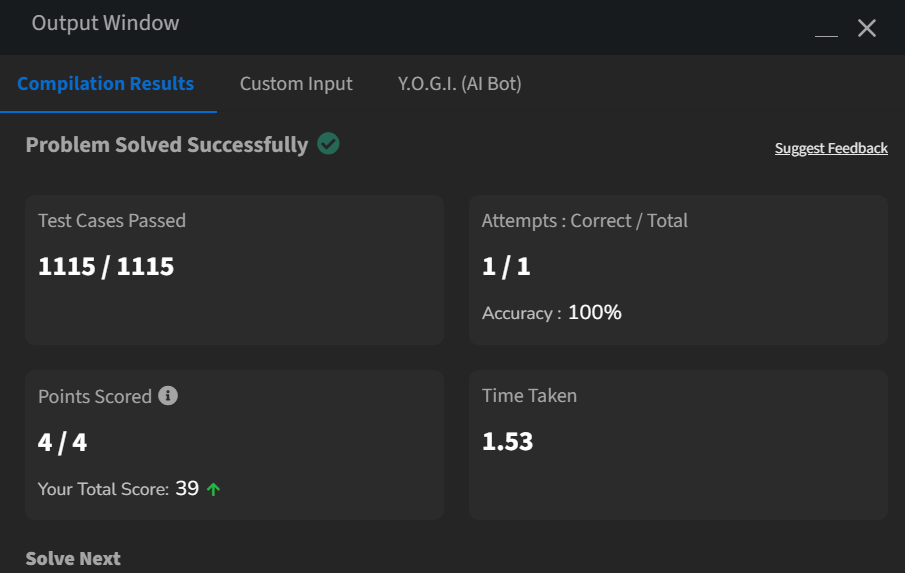
pq.add(i);

if(pq.size()>k) pq.poll();

} return pq.peek();

}

}



**Find the largest number**

Difficulty: **Medium**Accuracy: **40.7%**Submissions: **7K+**Points: **4**

Given an integer **N** the task is to find the largest number which is smaller or equal to it and has its digits in non-decreasing order.

**Examples 1:**

**Input:**

N = 200

**Output:**

199

**Explanation:**

If the given number

is 200, the largest

number which is smaller

or equal to it having

digits in non decreasing

order is 199.

class Solution{

static int find(int N){

char[] digits = Integer.toString(N).toCharArray();

int n = digits.length;

int mark = n;

for (int i = n - 1; i > 0; i--) {

if (digits[i] < digits[i - 1]) {

mark = i - 1;

digits[i - 1]--;

}

}

for (int i = mark + 1; i < n; i++) {

digits[i] = '9';

}

return Integer.parseInt(new String(digits));

}

}