**Name: Abishek S H Date:18/11/2024**

**Dept: CSE**

**1.Merge sort**

**Code:**

class Solution {

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

public static void bubbleSort(int arr[]) {

int temp;

int n=arr.length;

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

for(int j=0;j<n-i-1;j++){

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

temp=arr[j];

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

arr[j+1]=temp;

}

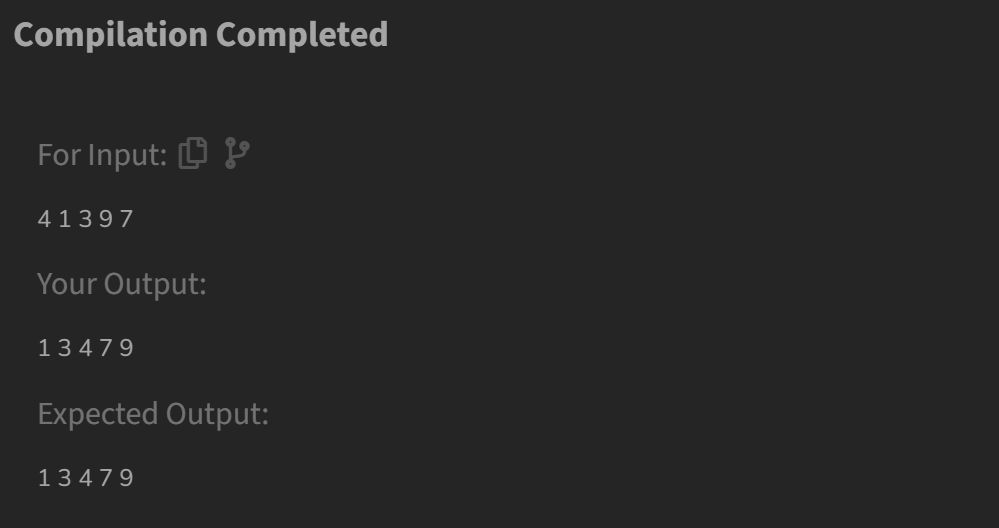
}

}

}

}

**Output:**



**Time complexity:** O(n^2)

**2.Quick Sort**

**Code:**

class Solution {

static void quickSort(int arr[], int low, int high) {

if (low < high) {

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

static int partition(int arr[], int low, int high) {

int pivot = arr[high];

int i = low - 1;

for (int j = low; j <= high - 1; j++) {

if (arr[j] < pivot) {

i++;

swap(arr, i, j);

}

}

swap(arr, i + 1, high);

return i + 1;

}

static void swap(int[] arr, int i, int j) {

int temp = arr[i];

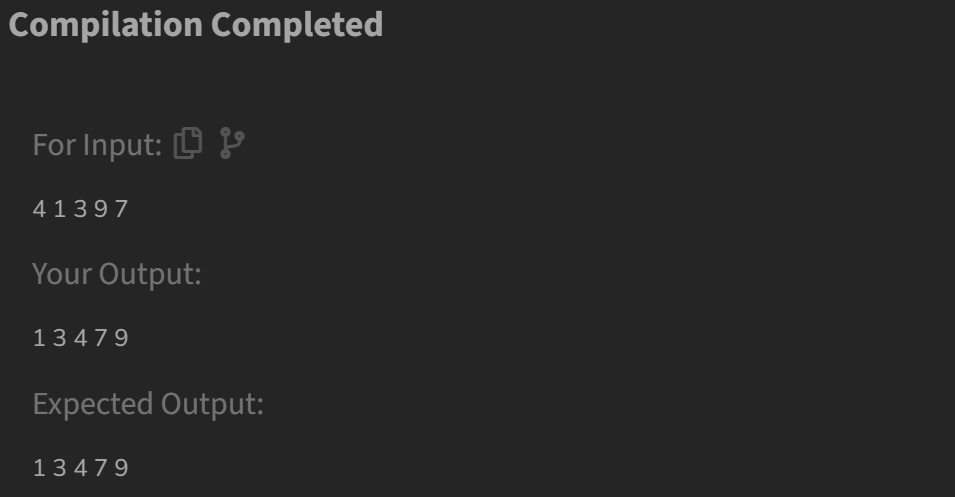
arr[i] = arr[j];

arr[j] = temp;

}

}

**Output:**

****

**Time Complexity: O(n log n)**

**3. First Non repeating character**

**Code:**

//{ Driver Code Starts

// Initial Template for Java

import java.io.\*;

import java.lang.\*;

import java.util.\*;

class Driverclass {

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

int t = sc.nextInt();

while (t-- > 0) {

String st = sc.next();

char ans = new Solution().nonRepeatingChar(st);

if (ans != '$')

System.out.println(ans);

else

System.out.println(-1);

System.out.println("~");

}

}

}

// } Driver Code Ends

// User function Template for Java

class Solution {

static char nonRepeatingChar(String s) {

Map<Character, Integer> hm = new LinkedHashMap<>();

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

char c = s.charAt(i);

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

}

for (Map.Entry<Character, Integer> entry : hm.entrySet()) {

if (entry.getValue() == 1) {

return entry.getKey();

}

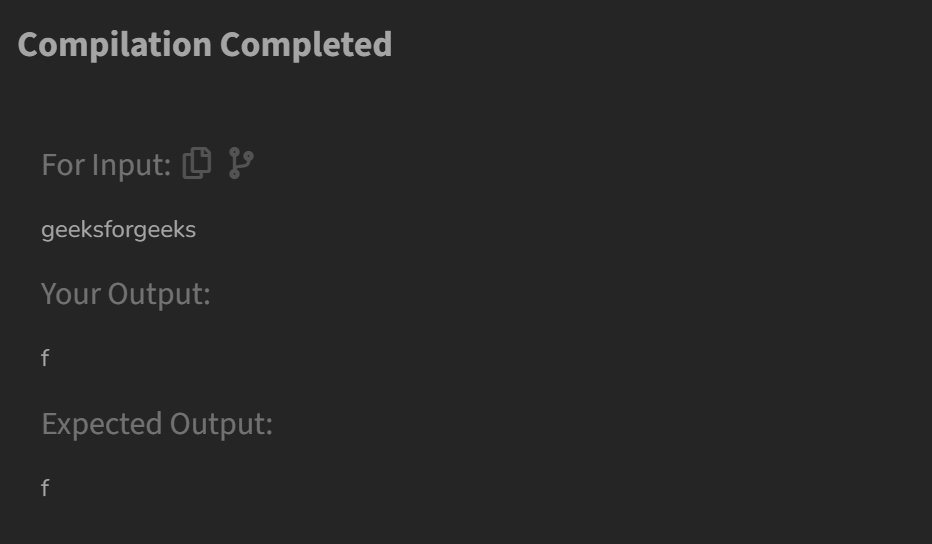
}

return '$';

}

}

**Output:**



**Time Complexity: O(n)**

**4. Edit Distance**

**Code:**

class Solution {

int minDisRec(String s1,String s2,int m,int n,int[][] memo ){

if (m == 0) return n;

if (n == 0) return m;

if (memo[m][n] != -1) return memo[m][n];

if (s1.charAt(m - 1) == s2.charAt(n - 1)) {

memo[m][n] = minDisRec(s1, s2, m - 1, n - 1, memo);

} else {

int insert = minDisRec(s1, s2, m, n - 1, memo);

int remove = minDisRec(s1, s2, m - 1, n, memo);

int replace = minDisRec(s1, s2, m - 1, n - 1, memo);

memo[m][n] = 1 + Math.min(insert, Math.min(remove, replace));

}

return memo[m][n];

}

public int editDistance(String s1, String s2) {

int m = s1.length(), n = s2.length();

int[][] memo = new int[m + 1][n + 1];

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

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

memo[i][j] = -1;

}

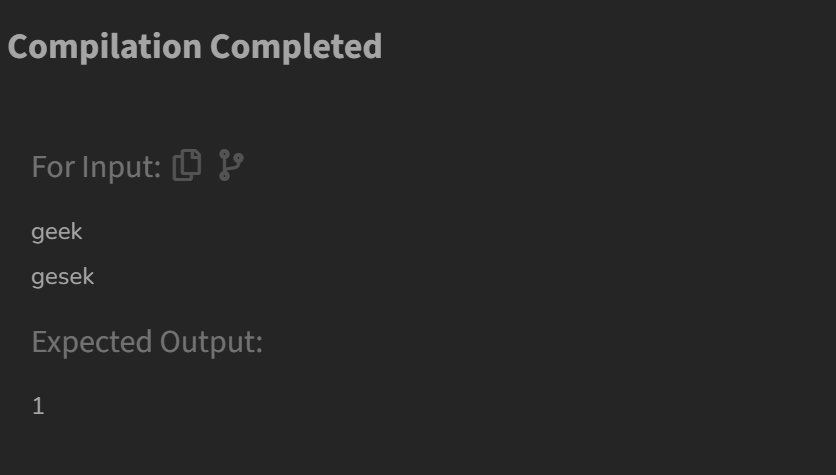
}

return minDisRec(s1, s2, m, n, memo);

}

}

**Output:**

****

**Time Complexity: O(m\*n)**

**5.K largest elements**

**Code:**

class Solution {

static List<Integer> kLargest(int arr[], int k) {

int n = arr.length;

Arrays.sort(arr);

List<Integer> res=new ArrayList<>();

int j=arr.length-1;

while(k!=0){

res.add(arr[j]);

j--;

k--;

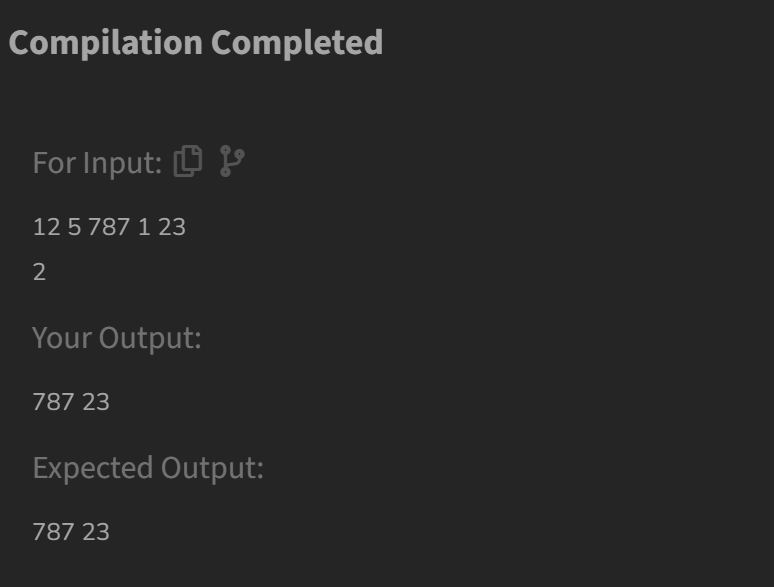
}

return res;

}

}

**Output:**

****

**Time Complexity: O(n log k)**

**6.Form the largest number**

**Code:**

class Solution {

String printLargest(int[] arr) {

String[] str = Arrays.stream(arr)

.mapToObj(String::valueOf)

.toArray(String[]::new);

Arrays.sort(str, (a, b) -> (b + a).compareTo(a + b));

if (str[0].equals("0")) {

return "0";

}

StringBuilder result = new StringBuilder();

for (String num : str) {

result.append(num);

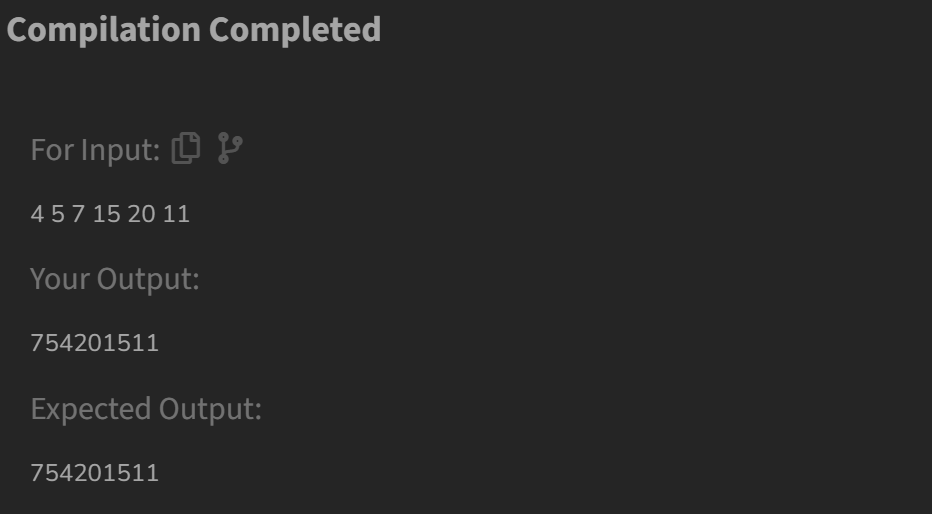
}

return result.toString();

}

}

**Output:**

****

**Time complexity: O(n log n)**