# Assignment 5

## Student Name: Aryan Mehta UID:22BCS12209

## Branch: BE-CSE (General) Section/Group: FL\_IOT-602 A

## Semester:6th Date of Performance: 04-03-25

## Subject Name: Advanced Programming Lab-2 Subject Code: 22CSP-351

# Aim: 389.Find the difference

# Implementation/ Code:

# class Solution {

# public char findTheDifference(String s, String t) {

# int sumS = 0, sumT = 0;

# for (char c : s.toCharArray()) sumS += c;

# for (char c : t.toCharArray()) sumT += c;

# return (char) (sumT - sumS);

# }

# }

# Output:

# 

# Aim: 976.Largest Perimeter Triangle

# Implementation/ Code:

# class Solution {

# public int largestPerimeter(int[] nums) {

# Arrays.sort(nums);

# for (int i = nums.length - 1; i >= 2; i--) {

# if (nums[i] < nums[i - 1] + nums[i - 2]) {

# return nums[i] + nums[i - 1] + nums[i - 2];

# }

# }

# return 0;

# }

# }

# Output:

# 

# Aim: 414.Third Maximum Number

# Implementation/ Code:

# class Solution {

# public int thirdMax(int[] nums) {

# TreeSet<Integer> set = new TreeSet<>();

# for (int num : nums) {

# set.add(num);

# if (set.size() > 3) set.remove(set.first());

# }

# return set.size() == 3 ? set.first() : set.last();

# }

# }

# Output:

# 

# Aim: 451.Sort Characters By Frequency

# Implementation/ Code:

# class Solution {

# public String frequencySort(String s) {

# Map<Character, Integer> freqMap = new HashMap<>();

# for (char c : s.toCharArray()) {

# freqMap.put(c, freqMap.getOrDefault(c, 0) + 1);

# }

# PriorityQueue<Map.Entry<Character, Integer>> maxHeap =

# new PriorityQueue<>((a, b) -> b.getValue() - a.getValue());

# maxHeap.addAll(freqMap.entrySet());

# StringBuilder result = new StringBuilder();

# 

# while (!maxHeap.isEmpty()) {

# Map.Entry<Character, Integer> entry = maxHeap.poll();

# result.append(String.valueOf(entry.getKey()).repeat(entry.getValue()));

# }

# return result.toString();

# }

# }

# Output:

# 

# Aim: 452.Minimum Number of Arrows to Burst Balloons

# Implementation/ Code:

# import java.util.Arrays;

# class Solution {

# public int findMinArrowShots(int[][] points) {

# if (points.length == 0) return 0;

# Arrays.sort(points, (a, b) -> Integer.compare(a[1], b[1]));

# int arrows = 1;

# int end = points[0][1];

# for (int i = 1; i < points.length; i++) {

# if (points[i][0] > end) {

# arrows++;

# end = points[i][1];

# }

# }

# return arrows;

# }

# }

# Output:

# 

# Aim: 881.Boats to Save People

# Implementation/ Code:

# class Solution {

# public int numRescueBoats(int[] people, int limit) {

# Arrays.sort(people);

# int left = 0, right = people.length - 1, boats = 0;

# 

# while (left <= right) {

# if (people[left] + people[right] <= limit) {

# left++;

# }

# right--;

# boats++;

# }

# return boats;

# }

# }

# Output:

# 

# Aim: 973.K Closest Points to Origin

# Implementation/ Code:

# class Solution {

# public int[][] kClosest(int[][] points, int k) {

# PriorityQueue<int[]> maxHeap = new PriorityQueue<>((a, b) ->

# Integer.compare((b[0] \* b[0] + b[1] \* b[1]), (a[0] \* a[0] + a[1] \* a[1]))

# );

# for (int[] point : points) {

# maxHeap.offer(point);

# if (maxHeap.size() > k) {

# maxHeap.poll();

# }

# }

# int[][] result = new int[k][2];

# for (int i = 0; i < k; i++) {

# result[i] = maxHeap.poll();

# }

# return result;

# }

# }

# Output:

# 

# Aim: 1338.Reduce Array Size to The Half

# Implementation/ Code:

# class Solution {

# public int minSetSize(int[] arr) {

# Map<Integer, Integer> freqMap = new HashMap<>();

# for (int num : arr) {

# freqMap.put(num, freqMap.getOrDefault(num, 0) + 1);

# }

# PriorityQueue<Integer> maxHeap = new PriorityQueue<>((a, b) -> b - a);

# maxHeap.addAll(freqMap.values());

# int removed = 0, count = 0, halfSize = arr.length / 2;

# while (removed < halfSize) {

# removed += maxHeap.poll();

# count++;

# }

# return count;

# }

# }

# Output:

# 