# Assignment 5

## Student Name: Manish Kumar UID:22BCS10249

## 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:

# //389. Find the Difference

# class Solution {

# public:

# char findTheDifference(string s, string t) {

# unordered\_map<char,int>mpp;

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

# mpp[t[i]]++;

# }

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

# mpp[s[i]]--;

# }

# for(auto it:mpp){

# if(it.second>0){

# return it.first;

# }

# }

# return '0';

# }

# };

# Output:

# 

# Aim: 976. Largest Perimeter Triangle

# Implementation/ Code:

# //976. Largest Perimeter Triangle

# class Solution {

# public:

# int largestPerimeter(vector<int>& nums) {

# sort(nums.begin(),nums.end());

# for(int i=nums.size()-1;i>1;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:

# // 414. Third Maximum Number

# class Solution {

# public:

# int thirdMax(vector<int>& nums) {

# sort(nums.begin(),nums.end());

# int largest,seclargest,thirdlargest;

# largest= nums[0];

# seclargest=nums[0];

# thirdlargest=nums[0];

# for(int i=0;i<nums.size();i++){

# if(nums[i]>largest){

# thirdlargest=seclargest;

# seclargest=largest;

# largest=nums[i];

# }

# else if(nums[i]>seclargest && nums[i]<largest){

# thirdlargest=seclargest;

# seclargest=nums[i];

# }

# else if(nums[i]>thirdlargest && nums[i]<seclargest){

# thirdlargest=nums[i];

# }

# }

# return ((nums.size()<=2 || seclargest==thirdlargest)?largest:thirdlargest);

# }

# };

# Output:

# 

# 4.

# Implementation/ Code:

# // 451. Sort Characters By Frequency

# class Solution {

# public:

# static bool st(pair<char,int>& a,pair<char,int>& b)

# {

# if (a.second == b.second)  return a.first < b.first;

# return a.second > b.second;

# }

# string frequencySort(string s) {

# unordered\_map<char,int> mp;

# for(char c:s)

# {

# mp[c]++;

# }

# vector<pair<char,int>> arr(mp.begin(),mp.end());

# sort(arr.begin(),arr.end(),st);

# string s1;

# for(auto& it:arr)

# {

# for(int i=0;i<it.second;i++)

# {

# s1+=it.first;

# }

# }

# return s1;

# }

# };

# Output:

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

# Implementation/ Code:

# // 452. Minimum Number of Arrows to Burst Balloons

# class Solution {

# public:

# int findMinArrowShots(vector<vector<int>>& points) {

# std::sort(points.begin(), points.end(), [](const auto& a, const auto& b) {

# return a[0] < b[0];

# });

# int arrows = 1;

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

# for (size\_t i = 1; i < points.size(); ++i) {

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

# arrows++;

# end = points[i][1];

# } else {

# end = std::min(end, points[i][1]);

# }

# }

# return arrows;

# }

# };

# Output:

# 

# 6.Aim: 881. Boats to Save People

# Implementation/ Code:

# // 881. Boats to Save People

# class Solution {

# public:

# int numRescueBoats(vector<int>& people, int limit) {

# 

# // sort vector

# sort(people.begin(),people.end());

# 

# int i = 0, j = people.size() - 1,cnt = 0;

# 

# while(i <= j)

# {

# // lightest person + heaviest person sum <= limit

# // they can go together

# if(people[i] + people[j] <= limit)

# {

# ++i;

# --j;

# }

# // if sum is over the limit,

# // heaviest will go alone.

# else

# --j;

# 

# ++cnt;  // number of boats

# }

# 

# return cnt;

# 

# }

# // for github repository link go to my profile.

# };

# Output:

# 

# 7. // 973. K Closest Points to Origin

# class Solution {

# public:

# vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {

# priority\_queue<pair<int, vector<int>>> closestpoints; // max heap

# for(auto point:points)

# {

# int x\_cord = point[0];

# int y\_cord = point[1];

# int dist = x\_cord\*x\_cord+y\_cord\*y\_cord; // no need of sqrt

# if(closestpoints.size()<k)

# {

# closestpoints.push({dist,point});

# }

# else if(dist<closestpoints.top().first)

# {

# closestpoints.pop();

# closestpoints.push({dist,point});

# }

# }

# vector<vector<int>> resultClosePoints;

# while(k>0)

# {

# resultClosePoints.push\_back(closestpoints.top().second);

# closestpoints.pop();

# k--;

# }

# return resultClosePoints;

# }

# };

# Output

# 

# 8. // 1338. Reduce Array Size to The Half

# class Solution {

# public:

# int minSetSize(vector<int>& arr) {

# unordered\_map<int, int> cnt;

# for (int x : arr) ++cnt[x];

# 

# vector<int> frequencies;

# for (auto [\_, freq] : cnt) frequencies.push\_back(freq);

# sort(frequencies.begin(), frequencies.end());

# 

# int ans = 0, removed = 0, half = arr.size() / 2, i = frequencies.size() - 1;

# while (removed < half) {

# ans += 1;

# removed += frequencies[i--];

# }

# return ans;

# }

# };

# Output

# 