# Assignment 5

## Student Name: Diksha Pandey UID:22BCS10359

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

## Semester:6th Date of Performance: 4-03-25

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

# **Aim:** [389. Find the Difference](https://leetcode.com/problems/find-the-difference/)

# Implementation/ Code:

# class Solution {

# public:

# char findTheDifference(string s, string t) {

# std::unordered\_map<char, int> count;

# 

# for (char c : t) {

# count[c]++;

# }

# 

# for (char c : s) {

# count[c]--;

# if (count[c] == 0) {

# count.erase(c);

# }

# }

# 

# return count.begin()->first;

# }

# };Output:Screenshot 2025-03-04 at 11.02.43 PM.png

# **Aim:** [976. Largest Perimeter Triangle](https://leetcode.com/problems/largest-perimeter-triangle/)

# Implementation/ Code:

# class Solution {

# public:

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

# 

# sort(nums.begin(), nums.end(), greater<int>());

# int ans;

# 

# 

# for(int i = 0; i < nums.size()- 2; i++)

# {

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

# {

# ans = nums[i]+nums[i+1]+nums[i+2];

# return ans;

# }

# }

# 

# return 0;

# }

# };

# Output:

# Screenshot 2025-03-04 at 11.05.28 PM.png

# **Aim:** [414. Third Maximum Number](https://leetcode.com/problems/third-maximum-number/)

# Implementation/ Code:

# 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:Screenshot 2025-03-04 at 11.06.45 PM.png

# **Aim:** [451. Sort Characters By Frequency](https://leetcode.com/problems/sort-characters-by-frequency/)

# Implementation/ Code:

# 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:Screenshot 2025-03-04 at 11.08.14 PM.png

# **Aim:**[452. Minimum Number of Arrows to Burst Balloons](https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/)

# Implementation/ Code:

# class Solution {

# public:

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

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

# int lastpoint = p[0][1];

# int ans = 1;

# for(auto point : p) {

# if(point[0] > lastpoint) {

# ans++;

# lastpoint = point[1];

# }

# lastpoint = min(point[1],lastpoint);

# }

# return ans;

# }

# };

# Output:Screenshot 2025-03-04 at 11.09.23 PM.png

# **Aim:** [881. Boats to Save People](https://leetcode.com/problems/boats-to-save-people/)

# Implementation/ Code:

# class Solution {

# public:

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

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

# int i=0 , boats=0;

# int j=people.size()-1;

# while(i<=j){

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

# i++;

# j--;

# boats++;

# }

# else{

# boats++;

# j--;

# }

# }

# return boats;

# }

# };

# };

# Output:Screenshot 2025-03-04 at 11.10.31 PM.png

# **Aim:** [973. K Closest Points to Origin](https://leetcode.com/problems/k-closest-points-to-origin/)

# Implementation/ Code:

# class Solution {

# public:

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

# 

# vector<vector<int>> result(k);

# 

# priority\_queue<vector<int>> maxHeap;

# 

# for (auto& p : points) {

# int x = p[0], y = p[1];

# maxHeap.push({x\*x + y\*y, x, y});

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

# maxHeap.pop();

# }

# }

# 

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

# vector<int> top = maxHeap.top();

# maxHeap.pop();

# result[i] = {top[1], top[2]};

# }

# 

# return result;

# }

# };

# Output:Screenshot 2025-03-04 at 11.12.22 PM.png

# **Aim:** [1338. Reduce Array Size to The Half](https://leetcode.com/problems/reduce-array-size-to-the-half/)

# Implementation/ Code:

# class Solution {

# public:

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

# unordered\_map<int,int>h;

# for(int i = 0; i < arr.size(); i++) h[arr[i]]++;

# priority\_queue<int> pq;

# for(auto it: h) pq.push(it.second);

# int ans = 0, minus = 0;

# while(!pq.empty())

# {

# ans++;

# minus += pq.top();

# pq.pop();

# if(minus >= (arr.size()/2)) break;

# }

# return ans;

# }

# };

# Output:Screenshot 2025-03-04 at 11.13.35 PM.png