UCS301 Data Structures Lab

Assignment 7

1. Write a program to implement following sorting techniques:
   a. Selection Sort
   b. Insertion Sort
   c. Bubble Sort
   d. Merge Sort
   e. Quick Sort

2. A slightly improved selection sort – We know that selection sort algorithm takes the minimum on every pass on the array, and place it at its correct position. The idea is to take also the maximum on every pass and place it at its correct position. So in every pass, we keep track of both maximum and minimum and array becomes sorted from both ends. Implement this logic.

Additional Questions

1. **Majority Element**

Given an array nums of size n, return the majority element. The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array. (Link:https://leetcode.com/problems/majority-element/description/?envType=problem-list-v2&envId=sorting)

Example 1:	**Example 2:**
Input: nums = [3,2,3]	Input: nums = [2,2,1,1,1,2,2]
Output: 3	Output: 2

2. **Top K Frequent in Array**

Given a non-empty integer array arr[ ] of size n, find the top k elements which have the highest frequency in the array. Note: If two numbers have the same frequencies, then the larger number should be given more preference. (Link:https://www.geeksforgeeks.org/problems/top-k-frequent-elements-in-array/1?page=2&category=Sorting&sortBy=submissions)

Example 1:	**Example 2:**
Input: arr[] = [3, 1, 4, 4, 5, 2, 6, 1], k = 2 Output: [4, 1] Explanation: Frequency of 4 is 2 and frequency of 1 is 2, these two have the maximum frequency and 4 is larger than 1.	Input: arr[] = [7, 10, 11, 5, 2, 5, 5, 7, 11, 8, 9], k = 4 Output: [5, 11, 7, 10] Explanation: Frequency of 5 is 3, frequency of 11 is 2, frequency of 7 is 2, frequency of 10 is 1.

3. **Maximum Sum Combination**

You are given two integer arrays a[ ] and b[ ] of equal size. A sum combination is formed by adding one element from a[ ] and one from b[ ], using each index pair (i, j) at most once. Return the top k maximum sum combinations, sorted in non-increasing order.

(Link:https://www.geeksforgeeks.org/problems/maximum-sum-combination/1?page=2&category=Sorting&sortBy=submissions)

Example 1:	**Example 2:**
Input: a[] = [3, 2], b[] = [1, 4], k = 2 Output: [7, 6] Explanation: Possible sums: 3 + 1 = 4, 3 + 4 = 7, 2 + 1 = 3, 2 + 4 = 6, Top 2 sums are 7 and 6.	Input: a[] = [1, 4, 2, 3], b[] = [2, 5, 1, 6], k = 3 Output: [10, 9, 9] Explanation: The top 3 maximum possible sums are : 4 + 6 = 10, 3 + 6 = 9, and 4 + 5 = 9

4. **Sort even-placed in increasing and odd-placed in decreasing order**

We are given an array of n distinct numbers. The task is to sort all even-placed numbers in increasing and odd-placed numbers in decreasing order. The modified array should contain all sorted even-placed numbers followed by reverse sorted odd-placed numbers.

Note that the first element is considered as even placed because of its index 0.

(Link:https://www.geeksforgeeks.org/dsa/sort-even-placed-elements-increasing-odd-placed-decreasing-order/)

Example 1:	**Example 2:**
Input: arr[] = {0, 1, 2, 3, 4, 5, 6, 7} Output: arr[] = {0, 2, 4, 6, 7, 5, 3, 1} Explanation: Even-place elements : 0, 2, 4, 6 Odd-place elements : 1, 3, 5, 7 Even-place elements in increasing order : 0, 2, 4, 6 Odd-Place elements in decreasing order : 7, 5, 3, 1	Input: arr[] = {3, 1, 2, 4, 5, 9, 13, 14, 12} Output: {2, 3, 5, 12, 13, 14, 9, 4, 1} Explanation: Even-place elements : 3, 2, 5, 13, 12 Odd-place elements : 1, 4, 9, 14 Even-place elements in increasing order : 2, 3, 5, 12, 13 Odd-Place elements in decreasing order : 14, 9, 4, 1

5. **Maximum Ice Cream Bars**

It is a sweltering summer day, and a boy wants to buy some ice cream bars.
At the store, there are n ice cream bars. You are given an array costs of length n, where costs[i] is the price of the ith ice cream bar in coins. The boy initially has coins to spend, and he wants to buy as many ice cream bars as possible.
Note: The boy can buy the ice cream bars in any order.
Return the maximum number of ice cream bars the boy can buy with coins.
You must solve the problem by counting sort.
(Link: https://leetcode.com/problems/maximum-ice-cream-bars/description/?envType=problem-list-v2&envId=sorting)

Example 1:	**Example 2:**	**Example 3:**
Input: costs = [1,3,2,4,1], coins = 7 Output: 4 Explanation: The boy can buy ice cream bars at indices 0,1,2,4 for a total price of 1 + 3 + 2 + 1 = 7.	Input: costs = [10,6,8,7,7,8], coins = 5 Output: 0 Explanation: The boy cannot afford any of the ice cream bars.	Input: costs = [1,6,3,1,2,5], coins = 20 Output: 6 Explanation: The boy can buy all the ice cream bars for a total price of 1 + 6 + 3 + 1 + 2 + 5 = 18.