Lab 3

Problem 1

A university is ranking students based on their exam scores. The scores must be sorted in **descending order**, maintaining **stability** (students with the same score stay in their original order).

📌 **What is Stability in Sorting?**

- **Definition:** A sorting algorithm is **stable** if it preserves the relative order of equal elements after sorting.
- **Example:**
  - **Unsorted:** `[("Alice", 85), ("Charlie", 85)]`
  - **Stable Sort Output:** `[("Alice", 85), ("Charlie", 85)]` ✅ *(Bob remains before Charlie)*
  - **Unstable Sort Output:** `[("Charlie", 85), ("Alice", 85)]` ❌ *(Order changed, unfair in ranking systems)*

You are given a list of students with their scores:

```
students = [("Alice", 85), ("Bob", 90), ("Charlie", 85), ("David",
92), ("Eve", 90)]
```

Explain **step by step** how **Insertion Sort** sorts this list.

1. **Describe each iteration** of Insertion Sort, showing comparisons, shifts, and insertions.
2. **Show the list's state** after each pass.

Problem 2

You are managing an **e-commerce platform** and need to display the **top K cheapest products** to customers. Instead of sorting the entire product list, you only need to identify the **K lowest-priced items efficiently**.

You are given the following product prices:

```
prices = [299, 150, 89, 199, 49, 120]
```

For **K = 3**, the three cheapest products should be:

```
[49, 89, 120]
```

1. **Explain step by step** how Selection Sort is used to find the **K smallest prices** without sorting the entire list.
2. **Show the intermediate states** of the list after each iteration until the K smallest elements are selected.

Problem 3

A **restaurant review platform** wants to display customer ratings **in ascending order** to help users easily find the lowest-rated restaurants. However, the ratings are currently **unsorted**, and the platform is using **Bubble Sort** to process them.

You are given the following **restaurant ratings (out of 5 stars)**:

```
ratings = [4.5, 3.2, 5.0, 2.8, 4.0, 3.8]
```

After sorting, the ratings should appear as:

```
[2.8, 3.2, 3.8, 4.0, 4.5, 5.0]
```

1. **Explain step by step** how Bubble Sort sorts this list, showing each full pass and the swaps that occur.
2. How can **Bubble Sort be optimized** for a nearly sorted list? Analyze the time complexity.

Problem 4

Different sorting algorithms perform **differently** depending on the structure of the input data. Your task is to **analyze and compare** the behavior of sorting algorithms when applied to different types of input.

Given the following input cases:

1. **Nearly sorted array:** `[1, 2, 3, 4, 6, 5, 7, 8, 9, 10]`
2. **Completely reversed array:** `[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]`
3. **Array with many duplicate elements:** `[4, 2, 2, 8, 3, 3, 3, 7, 4, 2]`

Which **sorting algorithm (Bubble Sort, Selection Sort, Insertion Sort, Merge Sort)** performs **best and worst** for each input type? Explain why

---

Problem 5

An **e-commerce platform** processes thousands of orders daily. Each order contains a **unique order ID** and a **total purchase amount**. To generate financial reports efficiently, the orders need to be **sorted in ascending order by total purchase amount** before analysis.

Given a list of **unsorted orders**, you must choose and explain the steps of a **sorting algorithm with `O(n log n)` complexity** to efficiently organize the data. **Describe step by step** how the sorting algorithm processes the list.

Problem 6

You are developing a digital photo management system. Users can add new photos to their albums in two ways:

1. **A few new photos** are added to an already sorted album.
2. **A large batch of photos** is imported at once, often in random order.

Which sorting algorithm would be most suitable for **each scenario**, and why? Consider factors such as **time complexity, efficiency, and adaptability** to different input conditions.