Insertion Sort - Assignment (10 Problems)

These problems will help you apply and improve your understanding of **Insertion Sort**. Solve them and present your solutions for benchmarking! 🚀

---

1. Sorting Employee Records (Ascending by Salary, Then by Name)

Problem Statement:

You have a list of employees, where each employee is represented as a tuple **(name, salary)**.

1. Sort the employees in **ascending order of salary**.
2. If two employees have the same salary, sort them **alphabetically by name**.

Example:

Input:
employees = [("Alice", 50000), ("Bob", 60000), ("Charlie", 50000), ("David", 70000)]

Expected Output:
[('Alice', 50000), ('Charlie', 50000), ('Bob', 60000), ('David', 70000)]

---

2. Sort a List of Floating-Point Numbers

Problem Statement:

You are given a list of floating-point numbers. Implement an **Insertion Sort algorithm** to sort them **in ascending order**.

Example:

Input:
numbers = [3.14, 1.41, 2.71, 1.73, 4.67]

Expected Output:
[1.41, 1.73, 2.71, 3.14, 4.67]

---

3. Sort a List of Strings by Length (Then Alphabetically for Equal Lengths)

Problem Statement:

You are given a list of words. Sort them in **ascending order based on length**.

- If two words have the same length, **sort them alphabetically**.

Example:

Input:
words = ["apple", "banana", "kiwi", "pear", "grape"]

Expected Output:
["kiwi", "pear", "apple", "grape", "banana"]

---

4. Sort Even and Odd Numbers Separately

Problem Statement:

Given a list of integers, **sort even numbers in ascending order and odd numbers in descending order** while maintaining their relative positions.

Example:

Input:
numbers = [5, 2, 9, 8, 1, 6]

Expected Output:

[9, 2, 5, 6, 1, 8]

(**Odd numbers sorted descending: [9,5,1], Even numbers sorted ascending: [2,6,8]**)

---

5. Sort Words Based on Number of Vowels

Problem Statement:

Sort a list of words in **descending order based on the number of vowels** they contain.

- If two words have the same number of vowels, sort **alphabetically**.

Example:

Input:
words = ["banana", "apple", "grape", "orange"]

Expected Output:
["banana", "orange", "apple", "grape"]

---

6. Find the Kth Smallest Element Using Insertion Sort

Problem Statement:

Given a list of numbers and an integer k, find the k-th smallest element **after sorting the list using insertion sort**.

Example:

Input:
numbers = [7, 4, 6, 3, 9, 1]
k = 3

Expected Output:
3

(The sorted list is **[1,3,4,6,7,9]**, and the 3rd smallest element is **3**.)

---

7. Implement Insertion Sort in Descending Order

Problem Statement:

Modify the standard **Insertion Sort algorithm** to sort numbers **in descending order** instead of ascending.

Example:

Input:
numbers = [4, 2, 9, 1, 5]

Expected Output:
[9, 5, 4, 2, 1]

---

8. Sort a List of Dates (DD-MM-YYYY) in Ascending Order

Problem Statement:

Given a list of dates in the format **DD-MM-YYYY**, sort them **in ascending order** using Insertion Sort.

Example:

Input:
dates = ["12-03-2022", "25-12-2021", "01-01-2023", "19-07-2022"]

Expected Output:
["25-12-2021", "12-03-2022", "19-07-2022", "01-01-2023"]

---

9. Sort a List of Tuples Containing Name and Age

Problem Statement:

You are given a list of tuples containing a **name and age**.

- Sort by **age in ascending order**.
- If two people have the same age, **sort alphabetically by name**.

Example:

Input:
people = [("Alice", 25), ("Bob", 30), ("Charlie", 25), ("David", 40)]

Expected Output:
[("Alice", 25), ("Charlie", 25), ("Bob", 30), ("David", 40)]

---

10. Check if a List is Already Sorted Using Insertion Sort Approach

Problem Statement:

Write a function that **checks if a list is already sorted** in ascending order.

- If sorted, return `True`, otherwise return `False`.

Example:

Input 1:
numbers = [1, 2, 3, 4, 5]

Expected Output:
True

Input 2:

numbers = [3, 1, 4, 2, 5]

Expected Output:
False

Instructions for Submission

- Solve all 10 problems using **Insertion Sort**.
- Save each problem in a separate file (`problem1.py`, `problem2.py`, ...).
- Ensure your code is **well-commented** and follows **clean coding practices**.
- Submit your solutions, and I'll **benchmark your implementations for efficiency and correctness**.

Let me know if you need **hints or explanations** on any question. Good luck! 🚀