Sorting Algorithms Practice Questions

This worksheet contains conceptual and programming practice questions on **Insertion Sort**, **Selection Sort**, and **Bubble Sort**.

Conceptual Questions

1. Explain the main idea behind **Insertion Sort**. How does it decide where to insert an element?
2. In **Selection Sort**, why does the number of comparisons remain the same even if the input list is already sorted?
3. Compare **Bubble Sort** and **Insertion Sort** in terms of the number of swaps they perform in the best case.
4. What is the **time complexity** (best, average, worst) of each of the three sorting algorithms?
5. Which sorting algorithm is **adaptive** (performs faster on partially sorted data)? Explain why.
6. Why is **Bubble Sort** considered a "stable" sorting algorithm?
7. If you have to sort a nearly sorted array of size 1000, which algorithm would you choose among these three and why?
8. Describe how many passes (outer loop iterations) are required for each sorting algorithm to completely sort an array of size *n*.

Programming Practice

1. Write a Python (or C++/Java) function to implement **Insertion Sort**. Test it on the list: `arr = [9, 5, 1, 4, 3]`. Show the array after each pass.
2. Implement **Selection Sort** to sort the array `[29, 10, 14, 37, 13]`.
   Print the smallest element found in each iteration.
3. Write a **Bubble Sort** implementation and count:
   - (a) the total number of **comparisons**, and
   - (b) the total number of **swaps**.
     Use the input `[5, 1, 4, 2, 8]`.
4. Modify **Bubble Sort** to stop early if the array becomes sorted before completing all passes.
5. Write a function that takes an array and a sorting method name (`'insertion'`, `'selection'`, `'bubble'`) and sorts accordingly.