Quick Sort Practice Set

Basic Level

Q1. Dry run Quick Sort on the array:
`[10, 7, 8, 9, 1, 5]` (Pivot = last element).

Q2. Apply Quick Sort step-by-step on:
`[4, 2, 7, 3, 1]`.

Q3. Show how Quick Sort sorts the array:
`[20, 10, 80, 30, 90, 40, 50, 70]` using pivot = first element.

Q4. If the array = `[5, 4, 3, 2, 1]`, show the **worst-case dry run** for Quick Sort (pivot = last element).

Q5. Perform Quick Sort on `[15, 3, 2, 1, 9, 5, 7, 8, 6]` (pivot = middle element).

Intermediate Level

Q6. Explain why Quick Sort's performance depends on **pivot selection**. Give examples of good and bad pivot choices.

Q7. Sort the array `[50, 23, 9, 18, 61, 32]` using Quick Sort. Show each partition.

Q8. Prove with an example that Quick Sort can be **in-place** (i.e., does not need extra arrays).

Q9. Apply Quick Sort on the array `[9, 12, 3, 5, 14, 10, 10]`. How are duplicate elements handled?

Q10. Show how Quick Sort sorts `[100, 90, 80, 70, 60, 50]`. Which case is this?

Advanced / Hard Level

Q11. Given array `[29, 10, 14, 37, 13]`:

- Perform Quick Sort with **random pivot selection**.
- Show one possible sequence of steps.

Q12. Explain why Quick Sort is **not stable** with an example.

Q13. If Quick Sort is applied to a linked list instead of an array:

- How will the **partition step** change?
- Is it efficient?

Q14. Suppose Quick Sort is applied recursively on an array of size `n`.

- What is the maximum depth of recursion in the **best case** and in the **worst case**?

Q15. You have the array `[38, 27, 43, 3, 9, 82, 10]`.

- Apply Quick Sort step-by-step.
- Compare the **recursion tree** of Quick Sort vs Merge Sort for the same array.