Exercise 3: Sorting Customer Orders

Analysis:

- Compare the performance (time complexity) of Bubble Sort and Quick Sort.
- Discuss why Quick Sort is generally preferred over Bubble Sort.

Time Complexity Comparison

Sorting Algorithm	Best Case	Average Case	Worst Case	Space Complexity
Bubble sort	O(n)	O(n^2)	O(n^2)	O(1)
Quick sort	O(n log n)	O(n log n)	O(n^2)	log(n)

Explanation:

- **Bubble Sort:**
  - Simple but inefficient.
  - Compares adjacent elements and "bubbles" the largest to the end in each pass.
  - Even with minor unsorted data, it still performs many unnecessary swaps.
- **Quick Sort:**
  - Divide-and-conquer algorithm that partitions the array around a pivot.
  - Generally fast for large datasets.
  - Worst-case $O(n^2)$ occurs when pivot choice is poor (e.g., already sorted list without randomized pivot).

Why Quick Sort is Generally Preferred

Quick Sort Advantages:

Reason	Explanation
Much faster in practice	Even though worst-case is $O(n^2)$, good pivot strategies (e.g., randomized or median-of-three) keep performance close to O(n log n)
Efficient for large datasets	Handles thousands/millions of records efficiently.
Inplace sorting	Doesn't require extra space (unlike Merge Sort)
Better CPU cache performance	Sequential memory access makes it cache-friendly

Bubble Sort Limitations:

Reason	Explanation
Inefficient for large datasets	Time complexity of $O(n^2)$ makes it impractical
Many unnecessary swaps	Poor performance even on moderately sized data
Mostly educational	Good for teaching sorting concepts, not used in production systems

Conclusion

- **Quick Sort** is the clear choice for sorting customer orders due to its speed, scalability, and in-place nature.
- **Bubble Sort** is only suitable for **very small lists** or for **learning purposes**.