### Problem Statement:

Write a program for analysis of quick sort by using deterministic and randomizedvariant.

## Problem Overview

You are tasked with analyzing Quick Sort using both its deterministic and randomized variants. Quick Sort is a divide-and-conquer algorithm where an array is partitioned around a pivot, and the subarrays are sorted recursively.

- **Deterministic Quick Sort** uses a fixed pivot (e.g., always the last element of the array).
- **Randomized Quick Sort** selects a random pivot to help improve the average-case performance by reducing the likelihood of encountering worst-case scenarios.

## Solution Outline

We will implement both versions of Quick Sort and compare their performance. Here's how the process will look:

1. Deterministic Quick Sort: The last element of the array is chosen as the pivot.
2. Randomized Quick Sort: A random element is chosen as the pivot.

## Explanation of the Code

- **Deterministic Quick Sort**:
  - A partition function is used to place the pivot element in its correct position, and the array is divided into two subarrays (elements smaller and larger than the pivot).
  - The quicksort_deterministic or quicksortDeterministic function recursively sorts the subarrays.
- **Randomized Quick Sort**:
  - The main difference is the pivot selection, which is done randomly.
  - Randomizing the pivot helps reduce the chance of hitting worst-case time complexity, especially for sorted or nearly sorted arrays.

## Expected Output

For both versions (Deterministic and Randomized):

Sorted array: [10, 30, 40, 50, 70, 80, 90]

The time taken for sorting will vary depending on the input size and randomness of the pivot.

**Conclusion**: Analysis of quick sort is done by using deterministic and randomized variant.