**Exercise 4: Employee Management System**

**Scenario:**

**You are developing an employee management system for a company. Efficiently managing employee records is crucial.**

**Steps:**

1. **Understand Array Representation:**

* **Explain how arrays are represented in memory and their advantages.**

**Array Representation in Memory**

1. *Contiguous Memory Allocation:* Arrays are stored in consecutive memory locations, meaning all elements are placed next to one another in memory.
2. *Fixed Size:* The size of an array is determined when it is created and cannot be altered dynamically.
3. *Index-Based Access:* Elements in an array can be accessed directly using their index, allowing for constant time complexity O(1)O(1)O(1) for access operations.

**Advantages of Arrays**

1. *Fast Access:* O(1) time complexity for accessing elements by index.
2. *Memory Efficiency:* Arrays are memory efficient with low overhead compared to structures like linked lists.
3. *Simplicity:* Arrays are straightforward to use and understand, making them suitable for basic data storage needs.
4. **Setup:**
   * **Create a class Employee with attributes like employeeId, name, position, and salary.**
5. **Implementation:**
   * **Use an array to store employee records.**
   * **Implement methods to add, search, traverse, and delete employees in the array.**
6. **Analysis:**

* **Analyze the time complexity of each operation (add, search, traverse, delete).**
* **Add Operation:** O(1) if the array has available space.
* **Search Operation:** O(n) since it may require checking each element in the worst-case scenario.
* **Traverse Operation:** O(n) because it involves visiting every element in the array.
* **Delete Operation:** O(n) as it might require shifting elements to fill the space left by the deleted item.
* **Discuss the limitations of arrays and when to use them.**

1. **Fixed Size:** Arrays have a predetermined size, limiting flexibility. Once full, you cannot add more elements unless a larger array is created.
2. **Inefficient Insertions and Deletions:** Inserting or deleting elements in the middle of an array requires O(n) time complexity, making these operations inefficient.
3. **Space Wastage:** If the array is not fully utilized, there may be wasted space.

Arrays can be used when: -

1. The number of elements is known beforehand.
2. Fast access to individual elements is required.