**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.

***Ans:*** In computer memory, arrays are represented as a contiguous block of memory where each element in the array is stored next to the previous one. This allows for efficient indexing because we can access any element by calculating its memory address using its index. The memory address of the first element (the base address) plus the index times the size of each element gives the address of the desired element.

Advantages:  
 **Efficient Indexing**: Accessing an element by its index is O(1), making arrays suitable for applications requiring frequent access to elements.

**Memory Locality**: Elements are stored in contiguous memory locations, which benefits from caching mechanisms and can lead to performance improvements.

**Ease of Use**: Arrays provide a straightforward way to store multiple items of the same type, making them easy to work with for fixed-size collections

***Setup:***

Create a class Employee with attributes like employeeId, name, position, and salary.

Code: EmployeeMS.java

***Implementation:***

Use an array to store employee records.

Implement methods to add, search, traverse, and delete employees in the array.

Code: EmployeeMS.java

***Analysis:***

1. Analyse the time complexity of each operation (add, search, traverse, delete).

*Add Employee():*

Best Case: O(1) (Appending to the end of the list)

Worst Case: O(1) (Appending to the end of the list)

*Search Employee():*

Best Case: O(1) (First element is the target)

Worst Case: O(n) (Target is not in the list, or last element is the target)

*Traverse Employees():*

Best and Worst Case: O(n) (Need to visit each element once)

*Delete Employee():*

Best Case: O(1) (First element is the target)

Worst Case: O(n) (Target is not in the list, or last element is the target)

1. Discuss the limitations of arrays and when to use them.

*Fixed Size:* If we use a static array, we must define its size at creation, which can lead to wasted memory if it's too large or insufficient capacity if it's too small.

*Inefficient Deletion and Insertion:* Deleting or inserting elements (other than at the end) requires shifting elements, which is O(n) in the worst case.

*Contiguous Memory Requirement:* Arrays require a contiguous block of memory, which can be problematic for large arrays or in fragmented memory environments.