**Exercise 4: Employee Management System**

**Array Representation in Memory**

* **Contiguous Memory Allocation**: Arrays store elements in consecutive memory locations, enabling efficient, direct access. This setup enhances performance due to fast indexing and cache-friendly operations.
* **Indexing**: Access elements in constant time O(1) using indices. Address calculation is straightforward and efficient.
* **Advantages**: Arrays provide quick access, better cache performance, and simple iteration due to their sequential storage.

**Time Complexity of Operations**

* **Add Employee**: Adding at the end of an array is O(1), a constant-time operation with no need for shifting elements.
* **Search Employee**: Searching requires scanning the array, leading to O(n) time complexity, as it might involve checking each element.
* **Traverse Employees**: Visiting each element once results in O(n) time complexity, with linear time required for processing.
* **Delete Employee**: Deleting involves locating the element and shifting others, both operations taking O(n) time.

**Limitations of Arrays**

* **Fixed Size**: Arrays cannot resize dynamically, which can lead to inefficient memory usage or resizing issues.
* **Inefficient Insertions and Deletions**: Adding or removing elements requires shifting, resulting in O(n) time complexity.
* **Contiguous Memory Allocation**: Requires a large contiguous block of memory, which can lead to allocation challenges.
* **No Direct Support for Advanced Operations**: Arrays lack native support for dynamic resizing or flexible operations, needing additional logic for such features.

**When to Use Arrays**

* **Fixed-size Collections**: Ideal when the number of elements is constant and known, providing efficient access and simple storage.
* **Fast Access**: Suitable when quick, direct access to elements via indices is needed.
* **Simple Storage Requirements**: Best for straightforward storage needs without requiring dynamic changes or complex operations.