**ANSWER**

**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.

Arrays are a fundamental data structure in programming, represented in memory as a contiguous block of elements, each accessed via an index. Here's how they are typically represented and their advantages:

* **Contiguous Memory Allocation:** Arrays are stored in contiguous memory locations. This means that elements are placed next to each other in a sequential block of memory.
* **Indexing:** Each element in an array can be accessed directly using its index. The address of any element can be calculated using the base address and the index.
* **Fixed Size:** The size of an array is fixed at the time of creation. This means the number of elements an array can hold is determined when the array is initialized.
* **Advantages**:
  + **Constant-Time Access (O(1))**: Any element can be accessed directly via its index.
  + **Efficient Traversal**: Iterating through an array is straightforward and efficient.
  + **Memory Locality**: Contiguous memory allocation improves cache performance, leading to faster data access.

1. **Analysis:**

* **Analyze the time complexity of each operation (add, search, traverse, delete).**

**Time Complexity of Operations**

1. **Add Employee:**
   * Operation: Adding an element to the end of the array.
   * Time Complexity: O(1) (if the array is not full).
2. **Search Employee:**
   * Operation: Linear search through the array.
   * Time Complexity: O(n).
3. **Traverse Employees:**
   * Operation: Iterating through all elements of the array.
   * Time Complexity: O(n).
4. **Delete Employee:**
   * Operation: Linear search followed by shifting elements.
   * Time Complexity: O(n).
   * **Discuss the limitations of arrays and when to use them.**

* **Limitations of Arrays**

1. **Fixed Size:** The size of the array is fixed at the time of creation. This can lead to wasted space if the array is not fully utilized or the need for resizing if more space is needed.
2. **Cost of Shifting Elements:** Inserting or deleting elements requires shifting elements, which can be time-consuming (O(n)).
3. **Inefficient for Search:** Linear search is required to find an element, which is O(n) in time complexity.

* **When to Use Arrays**

1. When the number of elements is known in advance and does not change frequently.
2. When direct access to elements is required.
3. When the data set is small and the simplicity of implementation is a priority.
4. When memory overhead needs to be minimized.