**Exercise 4: Employee Management System**

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

Arrays are a core data structure that store multiple items in a sequence. In memory, they occupy a continuous block of addresses, and each element shares the same data type. Access to elements is done through indexing, which reflects their position.

**Benefits of using arrays include:**

* **Constant-time Access:** Since the location of each element is directly computable using its index and the array’s base address, access time is **O(1)**.
* **Efficient Memory Usage:** Arrays are compact and don’t require extra memory to manage elements, making them space-efficient.
* **Ease of Use:** Their simplicity makes arrays easy to implement and ideal for basic data storage tasks.

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

1. **Add:** Inserting a new element takes constant time **O(1)** as long as there's available space in the array. If resizing becomes necessary, it introduces extra overhead—but we’re assuming a fixed-size array here.
2. **Search:** Searching for an employee has a time complexity of **O(n)**, since each element might need to be examined until the target is found.
3. **Traverse:** Visiting all employees in the array requires **O(n)** time, as each element is accessed once.
4. **Delete:** Removing an employee also takes **O(n)** time—you first locate the element, then shift all subsequent elements to maintain order.

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

Arrays come with some limitations that affect performance and flexibility:

* **Fixed Capacity:** After you define an array’s size, it can’t grow. So if the number of employees increases beyond the array’s capacity, you’ll need to manually resize it—often by creating a new, larger array and copying the elements.
* **Costly Deletions:** Removing an item isn’t efficient; you must shift all subsequent elements to fill the gap, which takes time.
* **Linear Search Time:** Searching through an array typically requires scanning each element, resulting in **O(n)** time for large datasets.

Arrays must be preferred in the following scenarios -

* If the number of entries is predetermined and constant.
* When you need very fast, direct access to elements using their index.
* If the dataset is small and operations like searching, adding, or deleting are rare or not time-critical.