# Exercise 4: Employee Management System

## 1. Understanding Array Representation:

- Arrays are contiguous blocks of memory where elements are stored at sequential memory locations.

- Advantages: Fast access via index (O(1)), simple structure, and minimal overhead.

- Ideal when the number of elements is known and fixed or relatively small.

## 2. Setup:

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

```
public class Employee {
  private int employeeId;
  private String name;
  private String position;
  private double salary;

  public Employee(int employeeId, String name, String position, double salary) {
    this.employeeId = employeeId;
    this.name = name;
    this.position = position;
    this.salary = salary;
  }

  public int getEmployeeId() { return employeeId; }
  public String getName() { return name; }
  public String getPosition() { return position; }
  public double getSalary() { return salary; }

  @Override
  public String toString() {
    return "Employee[ID=" + employeeId + ", Name=" + name +
        ", Position=" + position + ", Salary=" + salary + "]";
  }}
```

## 3. Implementation:

Using an array to store employee records and implement basic operations.

```
public class EmployeeManager {
    private Employee[] employees;
    private int size;

    public EmployeeManager(int capacity) {
        employees = new Employee[capacity];
        size = 0;
    }

    public void addEmployee(Employee emp) {
        if (size < employees.length) {
            employees[size++] = emp;
        } else {
            System.out.println("Employee array is full.");
        }
    }

    public Employee searchEmployee(int id) {
        for (int i = 0; i < size; i++) {
            if (employees[i].getEmployeeId() == id) {
                return employees[i];
            }
        }
        return null;
    }

    public void traverseEmployees() {
        for (int i = 0; i < size; i++) {
            System.out.println(employees[i]);
        }
    }

    public void deleteEmployee(int id) {
        for (int i = 0; i < size; i++) {
            if (employees[i].getEmployeeId() == id) {
                for (int j = i; j < size - 1; j++) {
                    employees[j] = employees[j + 1];
                }
                employees[--size] = null;
            break;
          }
        }
      }
    }
```

## 4. Analysis:

- Add: O(1) if within capacity.

- Search: O(n) as it may scan through all records.

- Traverse: O(n) to print all employees.

- Delete: O(n) due to shifting elements after deletion.

**Limitations of Arrays:**

- Fixed size; cannot dynamically expand.

- Insertion/deletion at arbitrary positions is inefficient (O(n)).

Arrays are suitable for small, fixed datasets with frequent reads but few inserts/deletes.