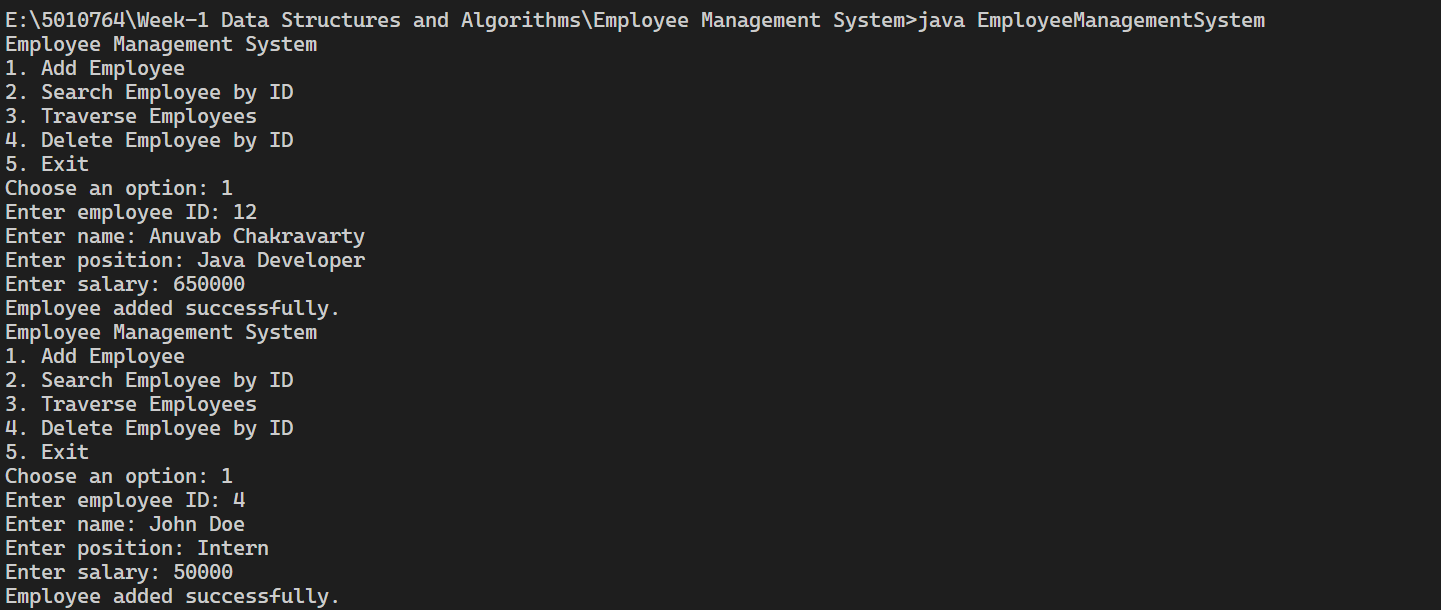
**EMPLOYEE MANAGEMENT SYSTEM:**

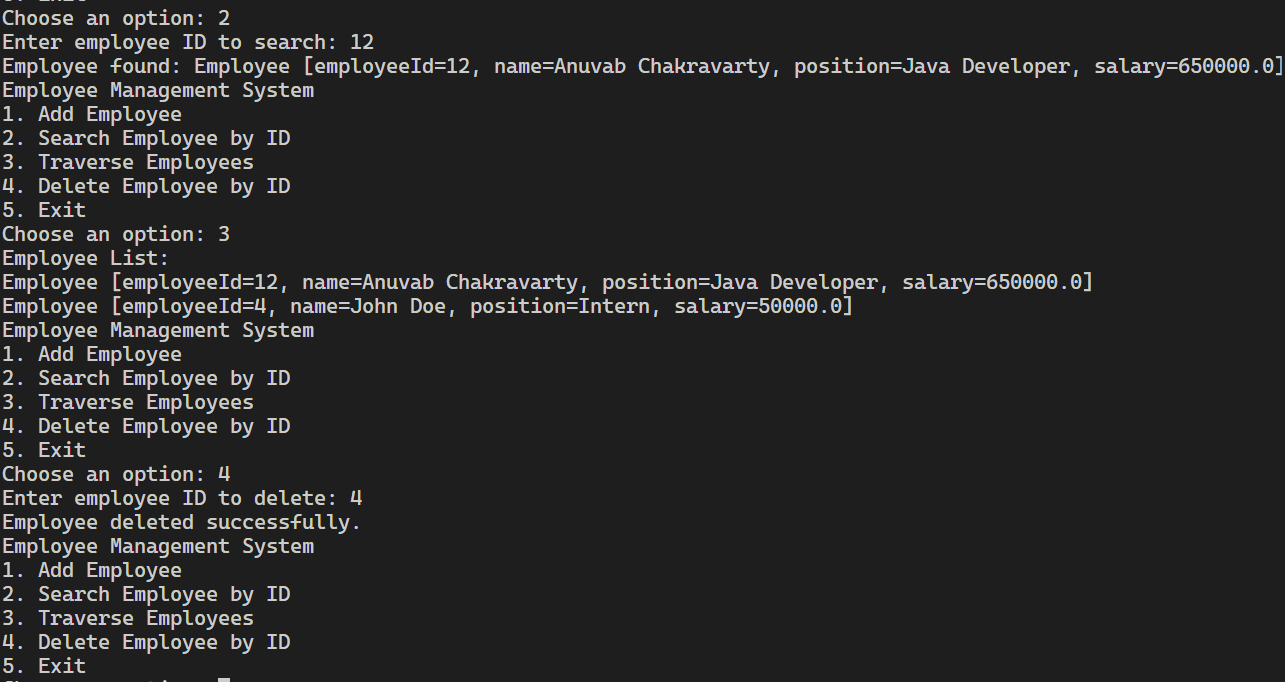
1. **Understanding Array Representation**
   * **Array Representation: Arrays are a fundamental data structure in memory that store elements in contiguous memory locations. This layout allows for efficient indexing and access to elements, as the position of each element can be directly computed from its index.**
   * **Advantages of Arrays:**
     + **Direct Access: Elements can be accessed in constant time, O(1), using their index.**
     + **Simplicity: Arrays are straightforward to implement and use, making them ideal for simple data management tasks.**
2. **Setup**
   * **Define an Employee class with attributes such as employeeId, name, position, and salary to represent each employee.**
3. **Implementation**
   * **Array Storage: Use an array to maintain employee records. Each element in the array will represent an Employee object.**
   * **Method Implementations:**
     + **Add Employee: Add a new Employee object to the array.**
     + **Search Employee: Locate an employee in the array based on their employeeId.**
     + **Traverse Employees: Iterate through the array to access each employee's details.**
     + **Delete Employee: Remove an employee from the array, which may involve shifting elements to fill the gap.**
4. **Analysis**
   * **Time Complexity Analysis:**
     + **Add Employee: O(1) if there is space in the array; otherwise, it could be O(n) if resizing is required.**
     + **Search Employee: O(n), as it requires scanning through the array to find the matching employee.**
     + **Traverse Employees: O(n), where n is the number of employees, as every element in the array is visited.**
     + **Delete Employee: O(n), due to the need to shift elements to fill the gap after deletion.**
   * **Limitations of Arrays:**
     + **Fixed Size: Arrays have a fixed size, which can be limiting if the number of employees changes frequently.**
     + **Inefficient Deletion and Insertion: Adding or removing elements requires shifting elements and can be inefficient in dynamic scenarios.**

**When to Use Arrays:**

* + **Arrays are suitable for scenarios where the number of elements is known and stable, and where quick access to elements via indexing is required. For more dynamic data where frequent insertions and deletions are expected, other data structures like lists or hash maps may be more appropriate.**

1. **Output**

****

****