**WEEK1\_ALGORITHMS\_DATA\_STRUCTURES**

**EXERCISE-4**

**Explanation of Array Representation**

**How Arrays are Represented in Memory:**

* **Contiguous Memory Allocation:** Arrays in Java are stored in contiguous memory locations. This means that each element in the array is stored next to the previous one, which allows for efficient indexing.
* **Fixed Size:** The size of an array is defined when it is created and cannot be changed dynamically.
* **Index-Based Access:** Elements in an array can be accessed directly using their index, which provides O(1) time complexity for access operations.

**Advantages:**

* **Fast Access:** Direct indexing allows for constant time access (O(1)) to elements.
* **Simplicity:** Easy to implement and manage due to fixed size and index-based access.

**Analysis**

**Time Complexity**

* **Add:** O(1) for adding if space is available. O(n) for resizing (not implemented here).
* **Search**: O(n) for linear search.
* **Traverse**: O(n) as it involves iterating over all elements.
* **Delete**: O(n) due to the need to shift elements after the deletion.

**Limitations of Arrays:**

* **Fixed Size**: Arrays have a fixed size, which can lead to wasted space if the number of employees changes.
* **Resizing**: Not straightforward if resizing is needed, leading to complexity in managing dynamic sizes.

**When to Use Arrays:**

* **Small or Fixed-Size Collections**: Arrays are suitable when the size of the collection is known and does not change frequently.
* **Simple Access and Operations**: Useful for simple use cases where the overhead of more complex data structures is not justified.