Q4. Understand Array Representation:

Arrays in Java are represented as contiguous blocks of memory. Each element in the array is stored in adjacent memory locations. This representation offers several advantages:

Fast access: Elements can be accessed directly using their index in O(1) time.

Memory efficiency: Arrays use a fixed amount of memory, which is allocated upfront.

Cache-friendly: Due to the contiguous memory allocation, arrays can benefit from CPU cache optimizations.

Analysis:

Time complexity of operations:

Add: O(1) amortized (occasionally O(n) when resizing is needed)

Search: O(n) in the worst case (linear search)

Traverse: O(n)

Delete: O(n) in the worst case (need to shift elements after deletion)

Limitations of arrays and when to use them:

Fixed size: Arrays have a fixed size once created. We've implemented a resizing mechanism, but it's costly (O(n)) when it occurs.

Inefficient insertions/deletions: Inserting or deleting elements in the middle of the array requires shifting other elements, which is O(n).

No built-in dynamic resizing: Unlike ArrayList, we have to manage resizing manually.

Arrays are best used when:

You know the maximum number of elements in advance.

You need fast, constant-time access to elements by index.

Memory usage is a concern, and you want to avoid the overhead of more complex data structures.

The data doesn't change size frequently.