Exercise 2:

Difference Between Array Size and Capacity

The size of an array refers to the number of elements currently stored in it, whereas the capacity represents the total number of elements the array can hold before requiring expansion. In dynamic arrays, the size increases as elements are added, but the capacity is typically larger to allow efficient resizing. The capacity is managed to reduce the frequency of memory reallocations, improving performance.

What Happens When an Array Needs to Grow Beyond Its Capacity?

When a dynamic array reaches its capacity, it must allocate a new memory block with a larger capacity to accommodate additional elements. If **there is available space** immediately following the array in memory, the system can expand it in place by increasing the capacity without moving elements. However, **if the memory after the array is occupied** by another variable, the system must allocate a new, larger memory block, copy the existing elements to the new location, and free the old memory. This process ensures continuous memory allocation but incurs a temporary overhead due to element copying.

Techniques to Amortize the Cost of Array Expansion

To minimize the overhead of frequent expansions, dynamic arrays often increase their capacity exponentially, commonly by doubling the size whenever the array is full. This technique ensures that insertions remain efficient on average, with most operations running in constant time O(1). Some implementations also use buffered growth strategies, allocating slightly more memory than needed to reduce the frequency of costly reallocations. These techniques help balance memory usage and performance, making dynamic arrays efficient for real-world applications.