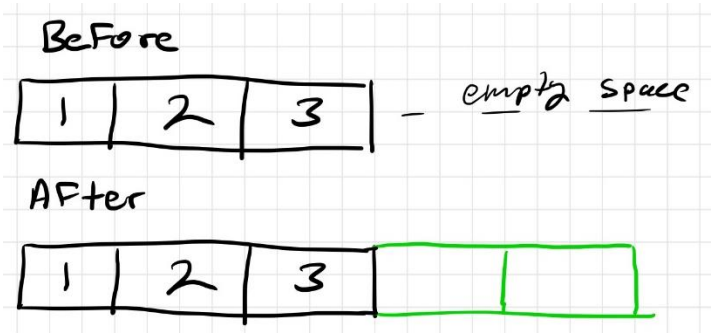
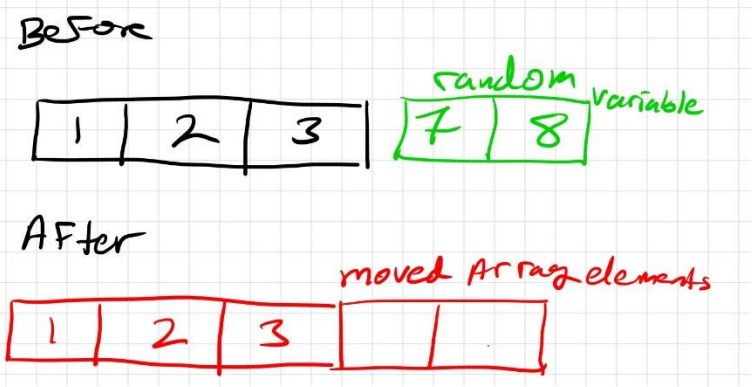
1. The size of an array refers to the number of elements that the array is currently holding. It's the count of actual data elements present in the array. The capacity of an array, on the other hand, refers to the total number of elements that the array can hold without resizing. It's the total amount of allocated memory for the array, which includes both occupied and unoccupied spaces.
2.
    1. The array can simply expand into the available space without any conflicts with other variables. Since the memory is available at the end of the array, this will be accomplished without any data movement. The process involves allocating the available memory block and updating the array's capacity accordingly.



    2. Expanding the array requires moving the existing array elements to a new memory location that has enough space to accommodate the expanded array. However, since there's another variable occupying the memory immediately after the array, it complicates the resizing process. The process involves finding a new contiguous memory block that can accommodate the expanded array and the other variable, copying the array elements and the other variable to the new location, updating the array's capacity, and adjusting references to the array.



3. Sure, real-world array implementations use several techniques to amortize the cost of array expansion. Here are a couple of them:
    1. Doubling the Size: One common technique is to double the size of the array each time it needs to be expanded. This means that each element ends up being copied over to a new array only a constant number of times. This strategy ensures that the time complexity of adding an element to the dynamic array is O(1) on average, even though individual operations might take more time. This is because the cost of resizing is spread out over the number of insertions.

2. Array Deque: Another technique is used in a data structure called an array deque (double-ended queue), which allows efficient addition and removal of elements at both ends. Instead of resizing the array when it's full, it maintains a front and back index and a fixed-size array. When an element is added or removed, it simply moves the appropriate index. When the array is full and an element needs to be added, it creates a new array of double the size and re-centers the data.