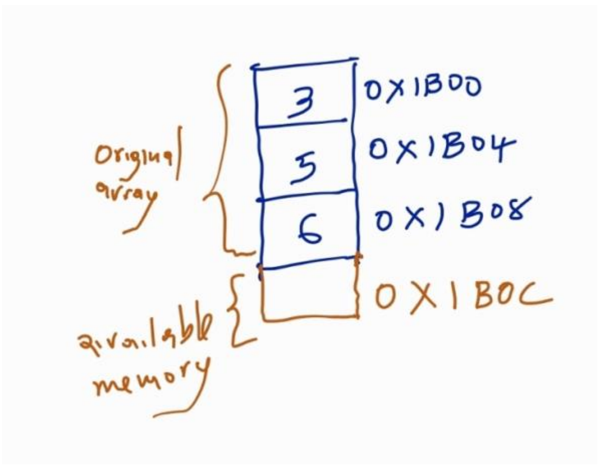
Exercise 2

1. **Difference Between Array Size and Capacity**
   Array size refers to the number of elements currently stored in the array, while capacity is the total number of elements the array can hold before needing to resize.

2. **What happen when an array needs to grow beyond its current capacity? Explain and produce a diagram showing the memory layout before and after expansion.**

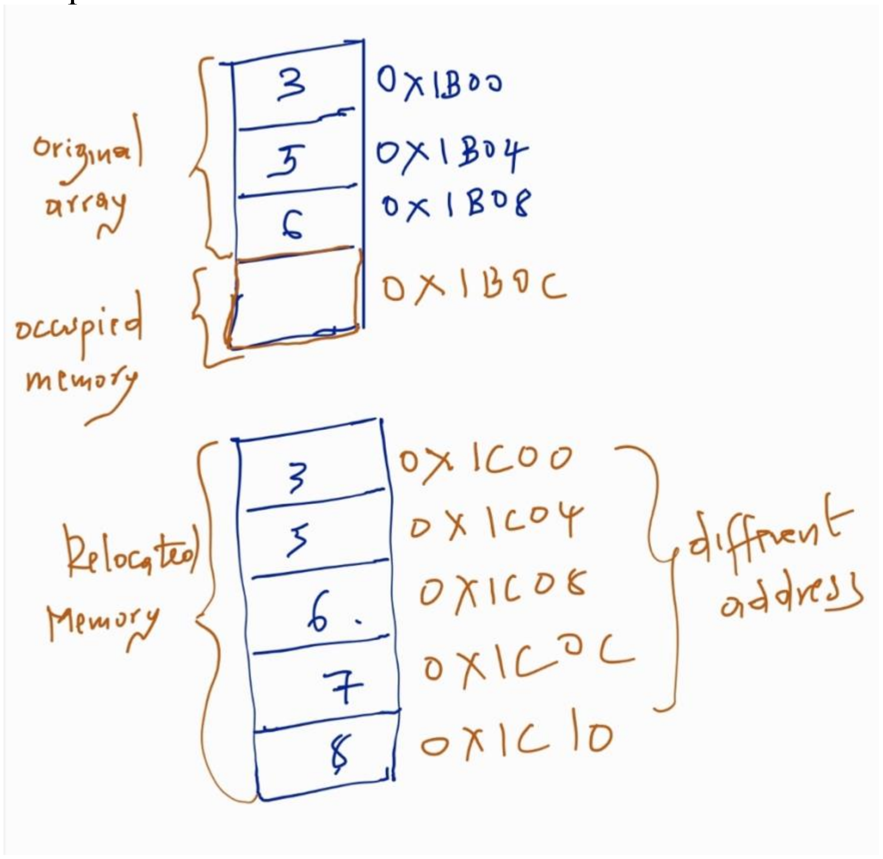
   1. First, consider the case where there is space in memory after the end of the array.
   If there is available space in memory immediately after the array, the operating system may allow the array to expand in place without needing to move the data. This is the most efficient scenario, as it avoids costly memory copying.



The Above array has free memory (0X1B0C) after the last element of the array which is at the address of 0x1B08. We can add a new element after the last element and increase the memory size by 1.

2. Then, consider the case where the memory after the end of the array is occupied by another variable. What happens in that case?

If the memory directly following the array is already allocated to another variable or process, the system must allocate a completely new memory block elsewhere, copy the existing elements into it, and then release the old memory. This operation can be expensive in terms of both time and computational resources.



The above array above has no space to contain a new element beyond the last element of the array which is at 0X1B0C. The entire array is relocated. Once the array has been relocated the array elements are copied and the new elements can be added for my illustration, i added two more elements 7 and 8.

3. Discuss one or more techniques real-world array implementations use to amortize the cost of array expansion.

- Geometric Growth Strategy: Rather than expanding the capacity by a fixed increment, many implementations scale it by a constant factor (e.g., 1.5x or 2x). This approach minimizes the frequency of resizing operations while maintaining an efficient balance between performance and memory consumption.

- Deferred Reallocation: Some systems postpone memory expansion until it becomes necessary, such as when an insertion fails due to insufficient space. This strategy prevents premature allocations, making it particularly useful in scenarios where future growth is unpredictable.