Exercise 2 — Dynamic Arrays (Group 12)

1. *Explain the difference between an array size and capacity [0.1 pts]*

The difference between array size and capacity is that array size corresponds to the number of elements in an array while capacity is the maximum number of elements that can be in the array. Array capacity is set as it uses up memory and allocates space to store the elements in the array.

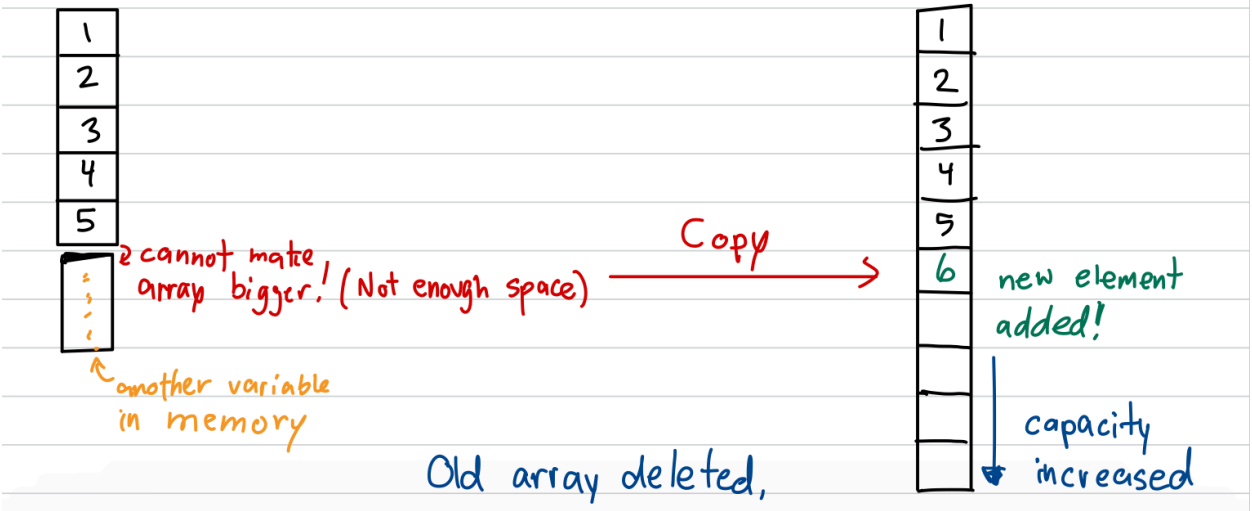
2. *What happens when an array needs to grow beyond its current capacity?*

   a. *First, consider the case where there is space in memory after the end of the array [0.1 pts]*

Element simply added to array in O(1) complexity with size value incremented.

   b. *Then, consider the case where the memory after the end of the array is occupied by another variable. What happens in that case? [0.2 pts]*

There is not enough memory to add the new element so the entire array must be copied to a place where there is enough memory to add a new element. Since an entire new array must be made, traversing through old array is required to copy all elements taking O(n) time.



If there is not enough space in memory to store the new element, then the entire array is copied into a new location where memory is available. Usually the array capacity doubles to fit the extra element. This process has an O(n) time complexity because copying the entire old array to the new one requires traversing through the old array until finally when the end is reached and the new element can be added. The old array is simply deleted from memory.

3. *Discuss one or more techniques real-world array implementations use to amortize the cost of array expansion [0.1 pts]*

As discussed earlier, when the array capacity needs to be increased due to it being full while trying to add an element, a growth factor of 2 is most often used meaning that the capacity of the array is multiplied by 2 in order to create enough space to add new elements which is beneficial because by doubling the array's size, the cost of copying elements becomes less frequent, and the amortized time complexity for each insertion remains O(1) on average.