ENSF 338 Lab 4

Group 35

Exercise 2

Explain the difference between an array size and capacity.

The array size is the total number of 'bytes' allocated to an array. For example, take an array like this,

```
array = new Array[10]
```

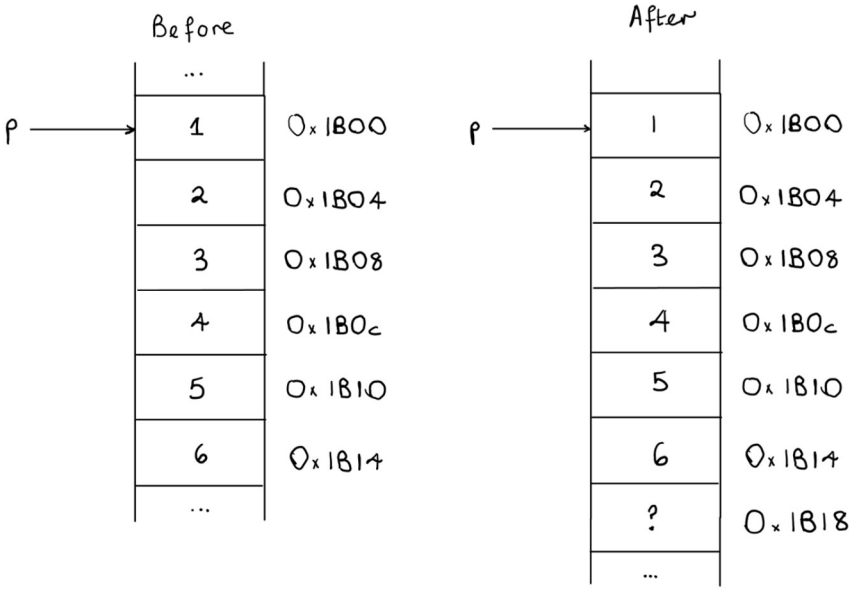
Assuming an array of integers, the size of the array will be,

```
10 * sizeof(int) = 40 bytes
```

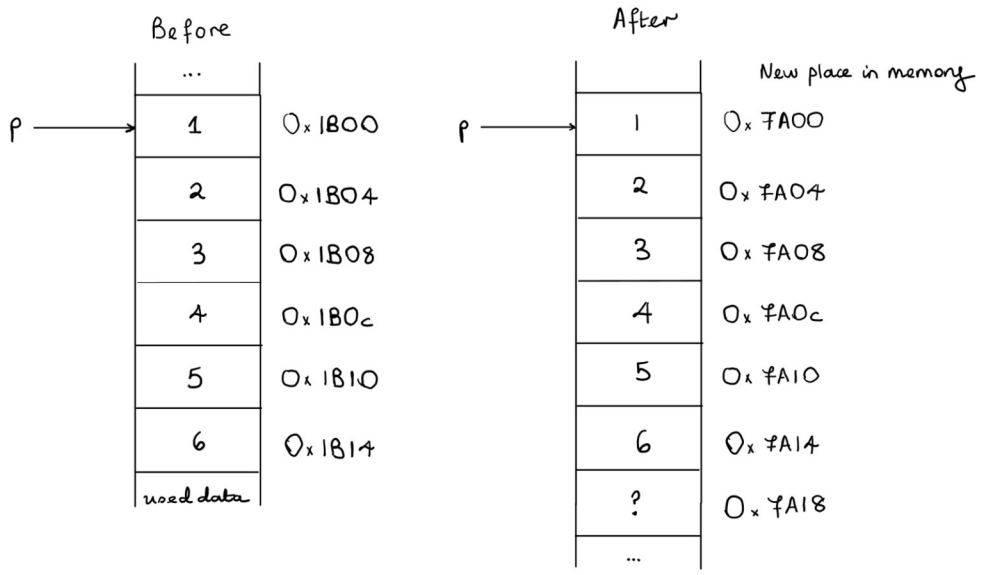
(even though the array is 'empty'). However, the capacity is the total number of elements that the array can contain. In the above example, the capacity is `10`.

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

Let's consider the case where there is space in memory after the end of the array. In this case, the size of the array is simply increased. That is, based on the data type of the array, more bytes are allocated at the end of the current array. In the diagram below, the array is expanded to include an additional element.



Now, let's consider the case where there is no space after the end of the array. In this case, the whole array will have to be relocated to a suitable place in memory. This procedure involves copying the whole array which takes $O(n)$ time. The diagram below shows this case.



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

A common implementation of array is 'vectors'. In this implementation, when the array needs to be expanded because it's reached its capacity, instead of just adding one element at a time, the capacity of the array is doubled. Essentially, if the current capacity is 10 and the array needs to be expanded, the new capacity will be 20. Since copying over the array is $O(n)$ and expensive to compute, this ensures that copying occurs less often.