Exercise 2:

1. ***Explain the difference between an array size and capacity***
   The size of an array refers to the number of elements actively being stored in the array at the current time. For a language like Python, *len(array)* would return the size of the array. Capacity, on the other hand, refers to the total amount of memory that has been allocated for the array. This indicates the total amount of elements that can be stored in the array before needing to allocate additional memory. Capacity is often larger than the size of the array in order to allow efficient resizing. In Python, for example, the growth factor of an array ranges from 1.125 to 1.5.

2. ***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***
   1. ***First, consider the case where there is space in memory after the end of the array***
      When an array needs to grow past its current capacity, granted that there is space in memory at the end of the array, Python will simply allocate the additional memory and append the elements as needed to the end of the array. This is an O(1) operation and, therefore, very efficient, but it must meet the requirement that there is space available in memory at the end of the array.

      Diagram Before Expansion:

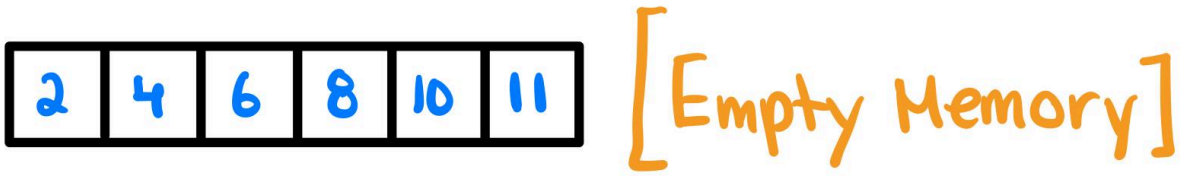
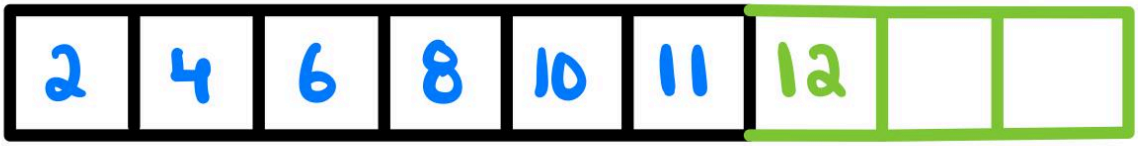
      

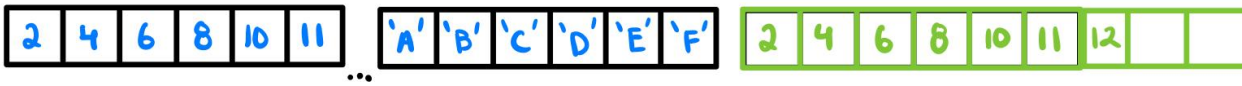
      Diagram After Expansion:

2. ***Then, consider the case where the memory after the end of the array is occupied by another variable. What happens in that case?***
   When an array needs to grow past its current capacity, but the memory at the end of the array is occupied by another variable, Python will allocate memory elsewhere for the entire array at a different location in memory. The existing array will then be moved one-by-one to the new memory location, and once complete, the appended element will be added to the end of the new array. Python will also automatically deallocate the memory for the original array location. Having to traverse the entire array and move each element to a new location causes this operation to have an O(n) time complexity.

   Diagram Before Expansion;

   Diagram After Expansion:



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

In a language Python, lists are implemented as dynamic arrays, which means that they use an over-allocation strategy in order to minimize the cost of resizing. When a list exceeds its current capacity, Python does not increase its capacity by only one to compensate the appended element, however, it increases the capacity by a growth factor of 1.125-1.5. This strategy of allocating more memory than needed results in less costly resizing operations that will be done in the future. This method of over-allocating memory is one way real-world array implementations amortize the cost of array expansion.