Iteration: Iteration refers to checking all the values index by index. The main idea is to go to that memory location and check all the values index by index.

```
def iteration(source):
  for i in range(len(source)):
    print(source[i])

def reverseIteration(source):
  for i in range(len(source) - 1, -1, -1):
    print(source[i])
```

Copy Array: Copy array means you initialize a new array with the same length as the given array to copy and then copy the old array's value by value. As the variable where we store the array only stores the memory location, only copying the value is not enough for array copying. For example, if you have an array titled arr = [1, 2, 3, 4] and write a2 = arr. It does not mean you have copied arr to a2.

```
def copyArray(source):
  newArray = [None] * len(source)
  for i in range(len(source)):
    newArray[i] = source[i]
  return newArray
```

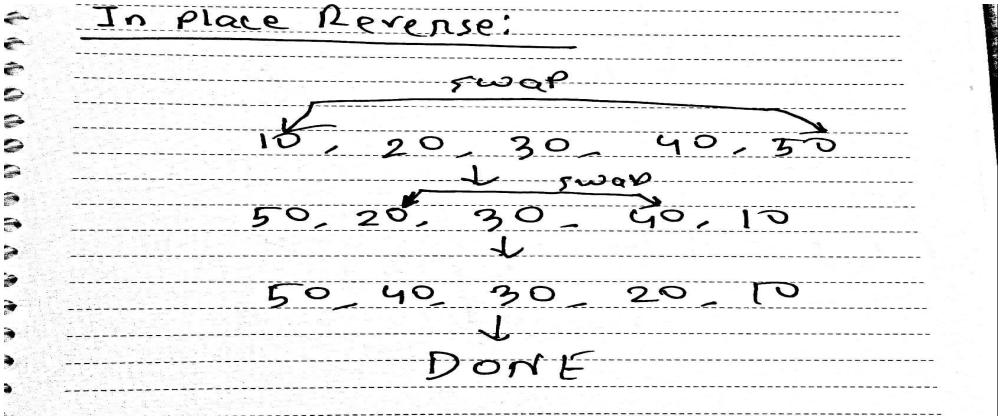
Resize Array: We can not resize an array because it has a fixed memory location. However, if we ever need to resize an array, we need to create a new array with a new length and then copy the values from the original array. For example, if we have an array [10, 20, 30, 40, 50] whose length is 5 and want to resize the array with length 8. The new array will be [10, 20, 30, 40, 50, None, None, None].

```python
def resizeArray(oldArray, newCapacity):
  newArray = [None] * newCapacity
  for i in range(len(oldArray)):
    newArray[i] = oldArray[i]
  return newArray
```

Reversing an Array: Reversing an array can be implemented in two ways. First, we will create a new array with the same size as the original array and then copy the values in reverse order. The method is called out-of-the-place operation.

```python
def reverseArrayOutOfPlace(arr):
  revArr = [None] * len(arr)
  i = 0
  j = len(arr) - 1
  while i < len(arr):
    revArr[i] = arr[j]
    i += 1
    j -= 1
  return revArr
```

However, an efficient approach might be to reverse the array in the original array. By this, we will not need to allocate extra spaces. This is known as an in-place operation. To do so we need to start swapping values from the beginning position to the end position. The idea is to swap starting value with the end value, then the second value with the second last value, and so on.




```python
def revArrInPlace(arr):
    i = 0
    j = len(arr) - 1
    while i < j:
        temp = arr[i]
        arr[i] = arr[j]
        arr[j] = temp
        i += 1
        j -= 1
```

Shifting an Array Left: Shifting an entire array left moves each element one (or more, depending how the shift amount) position to the left. Obviously, the first element in the array will fall off at the beginning and be lost forever. The last slot of the array before the shift (ie. the slot where the last element was until the shift) is now unused (we can put a None there to signify that). The size of the array remains the same however because the assumption is that you would something in the now-unused slot. For example, shifting the array [5, 3, 9, 13, 2] left by one position will result in the array [3, 9, 13, 2, None]. Note how the array[0] element with the value of 5 is now lost, and there is an empty slot at the end.

```
def shiftLeft(arr):
  for i in range(1, len(arr)):
    arr[i-1] = arr[i]
  arr[len(arr) - 1] = None
  return arr
```

Shifting an Array Right: Shifting an entire array right moves each element one (or more, depending how the shift amount) position to the right. Obviously, the last element in the array will fall off at the end and be lost forever. The first slot of the array before the shift (ie., the slot where the first element was until the shift) is now unused (we can put a None there to signify that). The size of the array remains the same however because the assumption is that you would something in the now-unused slot. For example, shifting the array [5, 3, 9, 13, 2] right by one position will result in the array [None, 5, 3, 9, 13]. Note how the array[4] element with the value of 2 is now lost, and there is an empty slot at the beginning.

```
def shiftRight(arr):
  for i in range(len(arr) - 1, 0, -1):
    arr[i] = arr[i - 1]
  arr[0] = None
  return arr
```

Rotating an Array Left: Rotating an array left is equivalent to shifting a circular or cyclic array left where the 1st element will not be lost, but rather move to the last slot. Rotating the array [5, 3, 9, 13, 2] left by one position will result in the array [3, 9, 13, 2, 5].

```
def rotateLeft(arr):
  temp = arr[0]
  for i in range(1, len(arr)):
    arr[i-1] = arr[i]
  arr[len(arr) - 1] = temp
  return arr
```

Rotating an Array Right: Rotating an array right is equivalent to shifting a circular or cyclic array right where the last element will not be lost, but rather move to the 1st slot. Rotating the array [5, 3, 9, 13, 2] right by one position will result in the array [2, 5, 3, 9, 13].

```
def rotateRight(arr):
  temp = arr[len(arr) - 1]
  for i in range(len(arr) - 1, 0, -1):
    arr[i] = arr[i - 1]
  arr[0] = temp
  return arr
```

Inserting an element into an Array: To insert an element into the array we need to make an empty slot first and then insert the value in the array. To make an empty slot first we need to check if any slots are available or not. If slots are not available then we need to resize the array using the concept of array resizing mentioned above. After that, we will use the idea of the right shift so that we can have an empty slot. The initialization point of the right shift would be the index where we want to insert the value. After the right shift operation, we will insert the value in that corresponding index.

For example, we have an array consisting of values [5, 7, 3, 6, None, None] (None represents the empty slots). Now if we want to insert 10 at index 1 we will need to use the right shift starting from index 1. So our array will be like [5, None, 7, 3, 6, None]. Here we can use the index 1 and insert the value 10 and the final array would be [5, 10, 7, 3, 6, None]

```python
def insertElement(arr, size, elem, index):
  # Practice how to throw exception if there is no empty space
  if size == len(arr):
    print("No space left. Insertion failed")
  else:
    for i in range(size, index, -1):
      arr[i] = arr[i - 1] #Shifting right till the index
    arr[index] = elem #Inserting element
    return arr
```

Removing an element from the Array: We need to use the concept of left shift to remove an element from the array. The idea is to start the left shift from the index which value we want to remove.

For example, we have an array consisting of values [10, 6, 8, 11, 15, 20] and want to remove value 8 which is at index 2, we need to do the left shift operation starting from index 2. So the resulting array would be [10, 6, 11, 15, 20, None].

```python
def removeElement(arr, index, size):
  for i in range(index + 1, size):
    arr[i - 1] = arr[i] #Shifting left from removing index
  arr[size - 1] = None #Making last space empty
```