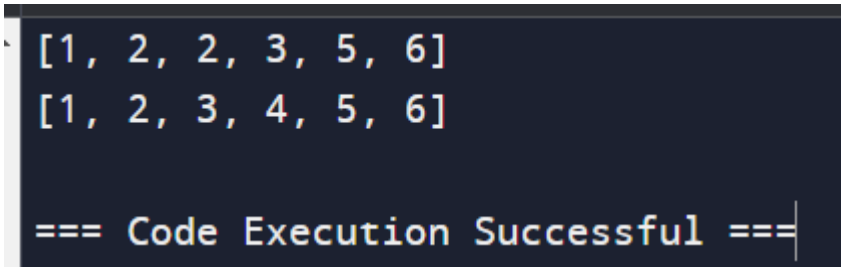
48. Merge Sorted Array You are given two integer arrays nums1 and nums2, sorted in non-decreasing order, and two integers m and n, representing the number of elements in nums1 and nums2 respectively. Merge nums1 and nums2 into a single array sorted in non-decreasing order. The final sorted array should not be returned by the function, but instead be stored inside the array nums1. To accommodate this, nums1 has a length of m + n, where the first m elements denote the elements that should be merged, and the last n elements are set to 0 and should be ignored. nums2 has a length of n. Example 1: Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3 Output: [1,2,2,3,5,6] Explanation: The arrays we are merging are [1,2,3] and [2,5,6]. The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1

PROGRAM:-

```
def merge(nums1, m, nums2, n):
    # Pointers for nums1 and nums2 respectively
    p1, p2 = m - 1, n - 1
    # Pointer for the last position in nums1
    p = m + n - 1

    # While there are still elements to compare
    while p1 >= 0 and p2 >= 0:
        if nums1[p1] > nums2[p2]:
            nums1[p] = nums1[p1]
            p1 -= 1
        else:
            nums1[p] = nums2[p2]
            p2 -= 1
        p -= 1

    # If there are still elements in nums2, add them
    nums1[:p2 + 1] = nums2[:p2 + 1]

# Example usage:
nums1 = [1, 2, 3, 0, 0, 0]
m = 3
nums2 = [2, 5, 6]
n = 3
merge(nums1, m, nums2, n)
print(nums1)  # Output: [1, 2, 2, 3, 5, 6]

# Another example:
nums1 = [4, 5, 6, 0, 0, 0]
m = 3
nums2 = [1, 2, 3]
n = 3
merge(nums1, m, nums2, n)
print(nums1)  # Output: [1, 2, 3, 4, 5, 6]
```

OUTPUT:-



TIME COMPLEXITY:-O((m+n)log (m+n))