Q12. Merge 2 sorted arrays in O(1) space.

i/p → arr1[] = {1, 3, 5, 7}
arr2[] = {0, 2, 6, 8, 9}
o/p → arr1[] = {0, 1, 2, 3}
arr2[] = {5, 6, 7, 8, 9}

Now here we need to merge 2 sorted arrays in the constant space. In merge sort, we merged arrays by taking 2 extra arrays. So let's jump on to the approach & dry run.

Let's say $i = n-1$ where n is the size of the 1st array and $j = 0$ where j is pointer to 0^{th} index of 2nd array & the size of 2nd array is m.
Just traverse until any one of the array finishes.
If while traversing we found arr1[i] > arr2[j], then we need to swap as arr1 will have lower elements whereas arr2 will have larger / bigger elements. Simply decrement i & increment j after the swapping is done.
Now if arr1[i] < arr2[j], we simply need to exit as for further i we won't get any i such that arr1[i] > arr2[j] as we are given sorted array, so simply break in the else part.

Dry run

1) arr1 → {1, 3, 5, 7} (i) arr2 → {0, 2, 6, 8, 9} (j)

arr1[i] > arr2[j] → simply swap & we get updated arrays. Also decrement i & increment j.

2) arr1 → {1, 3, 5, 0} arr2 → {7, 2, 6, 8, 9}

arr1[i] > arr2[j] → True & hence swap & then decrement i & increment j.

3) arr1 → {1, 3, 2, 0} arr2 → {7, 5, 6, 8, 9}

arr1[i] > arr2[j] → False & hence break.

Now final step is that sort both the arrays.

arr1 → {0, 1, 2, 3}
arr2 → {5, 6, 7, 8, 9}
Hence we have got the final answer.

Code

```
void merge (long long arr1[], long long arr2[],
            int n, int m) {

  int i = n-1; //Start from last index of arr1
  int j = 0; //Start from 0th index of arr2
  //Traverse until any one array or both finishes
  while (i>=0 && j<m) {
    if (arr1[i] > arr2[j]){
      swap(arr1[i], arr2[j]);
      i--; //Decrement i
      j++; //Increment j
    }
    else {
      break; //We are given sorted arrays.
    }
```

```
}
//Final step
sort (arr1, arr1 + n);
sort (arr2, arr2 + n);
}
```

↱ Final step sorting

Time complexity = $O((n+m)\log(n+m))$

Space complexity = $O(1)$