Practice: Merge sort, Quick sort

For all sorting algorithms: **Time and Space complexity. Stable? Adaptive? (Data moves)**

Algorithm	Stable	Adaptive	Time complexity			Space complexity
			Worst	Avg	Best	
Merge sort						
Quick sort						

Quick sort

QS1. Is quick sort stable?

If yes, prove it. If no, give an example array, A, (of size 5 or less), sort it with Quick_Sort, and indicate why it is not stable. Use the original array and the final, sorted array to base your proof (do not base your proof on a partially sorted array).

Hint: Focus on the pivot jump.

QS2. a) We make the call: `int res = partition(a, 0, 6);`

for each of the 2 example arrays **a** given in the table below. Give the arrays after the call, and the value returned in `res`. Use the partition method ~~from Cormen with LAST item~~ from class (and web Visual Algo) with **FIRST** item used as pivot.

	0	1	2	3	4	5	6	res
Original array **a** example 1	**10**	**6**	**12**	**8**	**15**	**7**	**13**	
Array after partition								
Original array **a** example 2	**9**	**11**	**12**	**16**	**3**	**8**	**17**	
Array example 2 after partition								

b) We make the call: `int res = partition(a, 0, 6);`

for each of the 2 example arrays **a** given in the table below. Give the arrays after the call, and the value returned in `res`. Use the partition method from web (Visual Algo) with **FIRST** item used as pivot.

	0	1	2	3	4	5	6	res
Original array **a** example 1	**13**	**7**	**12**	**8**	**6**	**15**	**10**	
Array after partition								
Original array **a** example 2	**17**	**11**	**12**	**16**	**3**	**8**	**9**	
Array example 2 after partition								

Merge sort

MS1. Show the array below after each call to the Merge (not Mergesort). Highlight the elements that are modified or "touched" by merge.

Index	0	1	2	3	4	5
Orig array	15	11	12	13	17	10