2) (5 pts) ALG (Sorting)

(a) (3 pts) Explain why, in the worst case, Quick Sort runs more slowly than Merge Sort.

In the worst case for Quick Sort, every time the array get split into two sides, if the split is extremely unequal (0 items on one size and all n-1 items except the partition element on the other side), worst case behavior occurs because there are n nested recursive calls on arrays of size n, n-1, n-2, and so forth.

In Merge Sort, it's guaranteed that the recursive calls always split into two arrays of roughly equal size. In general, the more equal the split between the two recursive calls is, the better the overall run-time will be. Because the Merge Sort split is essentially fixed, its worst case run time is near equal to its average case run time. But for Quick Sort, since this split at each level of recursion can be arbitrarily unequal, in the worst case where it's extremely unequal, the sort performs worse than Merge Sort.

Grading: The amount of detail above isn't necessary. Full credit to any response that recognizes that when making two recursive calls it's better to split the input array equally and that Merge Sort guarantees this but for Quick Sort this doesn't happen in the worst case.

Award partial credit as you see fit.

(b) (2 pts) In practice, Quick Sort runs slightly faster than Merge Sort. This is because the partition function can be run "in place" while the merge function can not. More clearly explain what it means to run the partition function "in place".

To run the partition function in place means that the function doesn't have to allocate significant extra memory other than the original array to sort that is passed to it. In particular, only a single temporary extra variable is needed to perform swapping (along with the usual loop index variables). Otherwise, most of the work occurs within the already allocated memory of the array passed to the partition function.

The merge function allocates a new array such that values from the original array are copied into the newly allocated array, then copied back to the original array. Thus, this function doesn't run in place as it routinely allocates a linear amount of memory (in the size of the arrays its merging) to perform its tasks. This runs slower in practice because of the extra copy back step, even though Merge Sort splits its data in the recursive step in a more equitable (and better) fashion.

Grading: Again, the amount of detail written above isn't necessary. Give full credit for any response that simply says that "in place" means performing the task without extra memory, (or a constant amount of extra memory.)

Award partial credit as you see fit.