prombblem 1.2 Sanford

2. In the standard MergeSort, each divide step splits the array into two parts, and hence the height of the recursion tree is $\log_2 n$. Then, merging takes $O(n)$ time per level.

For the modified algorithm where the array is split into thirds, the height of the recursion tree becomes $\log_3 n$ because we are dividing by three in each recursive call. There are more merge steps per level due to the tree-way merge, but beach step still takes $O(n)$ time since every element is considered once during the merge

The recurrence relation for the modified Merge Sort looks like this
$T(n) = 3T(\frac{n}{3}) + O(n)$

By the Master's theorm, this fits the case where $a=3$, $b=3$, and $f(n)=O(n)$. Here, $f(n)$ matches $n^{\log_b a} = n^{\log_3 3} = n$, which means that the work done at each level of recursion is linear, similar to the merge step in a regular Merge Sort.

Thus, the running time would still be $O(n \log n)$, but with a base of 3 for the logarithm, However, since logarithm bases can be changed with a constant factor (using the change of base formula), and we are ignoring constant factors and lower-order terms, the running time remains $O(n \log n)$

Thus the correct answer is b) $n \log n$