Problem 1 PDF

1. I chose to implement MergeSort. MergeSort is stable, but not in place. For example if we already have a sorted array, all that will happen is the algorithm will split every element into its own separate array, then merge them back together. In this process no elements will be moved out of position. MergeSort has time complexity guarantee of O(NlogN)
2. The best case of MergeSort is still O(NlogN) This is because regardless of the number of sorting elements the process is still the same. Even if the array is partly or fully sorted, the number of steps is not reduced.

**for** (i=0, i<= size-1, i++){

a[i]=b[i];

size=size/2;

partition(**int**[] b)}

**for**(i=r, i>size-1, i--){

a[i]=c[i];

size=size/2

partion(**int**[] c);

3) The worst case of MergeSort is O(NlogN). Even if we take the worst case and say the array is not sorted at all then the number of steps mergesort takes is still the same.**for** (i=0, i<= size-1, i++){

a[i]=b[i];

size=size/2;

partition(**int**[] b)}

**for**(i=r, i>size-1, i--){

a[i]=c[i];

size=size/2

partion(**int**[] c);