EC330 HW 4

Chase Monald
U18719879
B2

1) sortA (Array A [0...n-1])
```
for (i=0 to n/2)
    for (j=n/2+1 to n-1)
        if (A[i] <= A[j])
            swap(A[i], A[j])
return A;
```

[1, 2, 3, 4]

i.) When the algorithm sortA is run on the array [8, 7, 6, 5, 4, 3, 2, 1] n=8, thus the i loop is performed 5 times (0th-4th index). The j loop is performed 3 times (5th-7th index). Through these iterations, A[i] is never < or = to A[j]. In the last iteration, we are closest, where 4 <= 3. The result of the sort is the original vector, because no operations are performed; [8, 7, 6, 5, 4, 3, 2, 1].

ii.) For all arrays of size n ≥ 8 with only positive integers, you can assume the algorithm sortA probably does not sort the array. Repeatedly comparing two halves of an array in this way isn't logically sound for sorting. A counterexample

iii.) that sorts while splitting a vector/data set in half is merge sort. This has O(nlogn) runtime, sorts each half of the data independently, then merges the data & sorts the merged data recursively. Perhaps sortA would be a valid algorithm if it implemented a merge sort type merge & it sorted each half independently.