NAMES: John Karuntzos / Dakota K / Kristopher Hoce

Worksheet 26: Work with either ONE or TWO other students to complete this worksheet. You must work with at least one other student on this assignment, and should submit one paper. Be sure to describe your key idea at a high level.

1. Compare the performance of heap-sort on arrays that are sorted in ascending order to arrays that are sorted in descending order. Include a discussion of the number of operations involved in each part of the process of heap-sort: building the heap to have the max-heap property, interchanging the root and the last element in the heap, and the subsequent max-heapify process. Use specific examples of sorted lists in your discussion, and be complete. (6 points)

```
HeapSort(A):
    BuildMaxHeap(A)          //n/2 times if ascending order //n*n/2 if descending
    for i <— to A.length to 2:                                        //Ascending -> n^2
        Interchange A[1] and A[i]       //n                           //Descending -> 1/2 * n^4
        A.heapsize <— A.heapsize - 1
        MaxHeapify(A,1)                 //N if in ascending order //N^2 times if descending
```

```
BuildMaxHeap(A):
    A.heapsize <— A.length
    for i <— floor(A.length/2) to 1:
        maxHeapify(A,i)
```

```
MaxHeapify(A,i):
    L <— left(i)
    R <— right(i)
    if A[L] > A[i] AND L <= A.heapsize:
        largest <— L
    else:
        largest <— i
    if A[R] > A[largest] AND r <= A.heapsize:
        largest <— R
    if largest != i:
        exchange A[i] and A[largest]
        maxHeapify(A,i)
```

If an array is sorted in ascending order the runtime in terms of N elements in the list would be O(N^2). An array in descending order would have a runtime of O(N^4).
The MaxHeapify function runs n times for descending to compensate for the order. And it only runs once if already in Ascending.
The BuildMaxHeap runs n/2 * the maxHeapify runtime

EX: {1,2,3} => N^2 => 9
EX: {3,2,1} => 1/2 * N^4 => 41