1. My choice of algorithm was a insertion sort. I know that this is not O(Nlog(N)) time, however I was running out of time and aimed to get partial credit for a correct sort method rather than taking a 0 on the question. Had I had time, I would have done a merge sort algorithm on the list. My sort satisfies the relative ordering requirement because it moves smaller values backwards in the array until they are larger than the element directly prior to itself, executing on teach element in order. This ensures a correctly sorted method as a final product.
   1. **for** (**int** i=0; i<a.length-1; i++){  
       **for** (**int** j=i; j>=0; j--){  
       **if** (a[j] > a[j+1]){  
       **int** swap = a[j];  
       a[j] = a[j+1];a[j+1] = swap;  
       }  
       }  
      }
2. The best case complexity of my algorithm is O(N^2) time as the program will execute the entirety of both for loops no matter how well the array is sorted before the method is called.
3. The worst case complexity of my algorithm is O(N^2) time as the program will execute the entirety of both loops no matter how badly the array is sorted before the method is called.