Problem 1:

1) Here I ended up using selection sort, which actually performs the sort. I decided to use this sort as it seemed to be a simple algorithm that would start at the first array element and then find the min array element to its right. This is done in the inner for loop:

**for**(**int** i=0; i<size; i++)

{

min=i;

**for**(**int** j=i+1; j<size; j++)

{

**if**(a[j]<a[min])

{

min=j;

}

}

*swap*(a, min, i);

}

After the min element is found a swap is done which I used through a helper method called swap. This ends up being stable because, if given a sorted array already, the keys would just be swapped with itself, and the array will remain sorted. Therefore the algorithm is stable.

2) The best case time complexity here is O(n^2). This is due to the nested for loop shown above. The inner for loop runs n times for n elements which is O(n^2)

3) The worst case is O(n^2) as well. When observing the double for loop regardless of the best case or worst case, the inner loop runs n times for n elements which is O(n^2)