CSC 225 - LAB 5

In a sequence $S = [s_1, s_2, \ldots, s_n]$ of n integers, an *inversion* is a pair of elements s_i and s_j where $i < j$ (that is, s_i appears before s_j in the sequence) and $s_i > s_j$. For example, in the sequence

$$S = 2, 1, 5, 3, 4$$

the pairs (2,1), (5,3) and (5,4) are inversions.

An array with n elements may have as many as

$$\binom{n}{2} = \frac{n(n-1)}{2}$$

inversions. When the number of inversions, k, may be any value between 0 and $\frac{n(n-1)}{2}$, the best algorithm for counting inversions has running time $O(n \log n)$. There also exists a $O(n + k)$ algorithm for counting inversions, which is $O(n^2)$ when $k \in O(n^2)$.

Your goal in this lab is to create two algorithms which count the number of inversions in an input sequence:

Input: An array A of n integers in the range 1 to n.
Output: An integer, corresponding to the number of inversions in A.

The first algorithm will have a $O(n \log n)$ runtime and will be better for counting inversions when $k > n$. The second algorithm will have runtime $O(n + k)$ and will be better when $k \leq n$ (i.e. $O(n + n) = O(n)$).

Bonus:
Time permitting, you should try to implement them.