CSC 225 SUMMER 2016
ALGORITHMS AND DATA STRUCTURES I
ASSIGNMENT 2 - PROGRAMMING
UNIVERSITY OF VICTORIA

1 Programming Assignment

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.

The programming assignment is to implement an algorithm which counts the number of inversions in an input sequence:

Input: An array A of n integers in the range $1 - n$.

Output: An integer, corresponding to the number of inversions in A.

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 $k \in O(n^2)$.

In this assignment, we will assume that the number of inversions is at most n (that is, $k \leq n$). When the $O(n + k)$ algorithm is used with such values of k, its running time is $O(n + n) = O(n)$. For full marks, your implementation must run in $O(n)$ time when $k \leq n$.

Your task is to write a java program, stored in a file named `CountInversions.java` that contains a function `CountInversions`, which takes an integer array *A* as its only argument, and returns an integer value. The main function in your code should help you test your implementation by getting test data or reading it from a file. It should also handle errors but don't worry about running times.

2 Examples

The table below gives the inversion count and a list of the inversions present in several small input arrays.

Input Array	# of Inversions	Inversions
1, 2, 3	0	none
1, 2, 3, 5, 4	1	(5, 4)
10, 30, 40, 20, 50	3	(30, 20), (40, 20)
4, 3, 2, 1, 5, 6	6	(4, 3), (4, 2), (4, 1), (3, 2), (3, 1), (2, 1)
5, 1, 2, 3, 4	4	(5, 1), (5, 2), (5, 3), (5, 4)

3 Test Datasets

A set of input files have been uploaded to ConneX, containing arrays of various sizes and inversion counts. You should test your implementation on the uploaded files before submitting. Note that an algorithm with a running time which is $O(n \log n)$ or better should be able to process any of the files within a few seconds. The uploaded files may not cover all possible cases, so you should test your implementation on other inputs as well.

4 Evaluation Criteria

The programming assignment will be marked out of 25, based on a combination of automated testing (using large test arrays similar to the ones posted on ConneX) and human inspection. All of the tested input arrays will have an inversion count k which is at most n.

Score (/25)	Description
0 - 5	Submission does not compile or does not conform to the provided template.
5 - 15	The implemented algorithm is $O(n^2)$ or is substantially inaccurate on the tested inputs.
15 - 20	The implemented algorithm is $O(n \log n)$ or contains minor errors.
20 – 25	The implemented algorithm is $O(n + k)$ on an array with n elements and k inversions, resulting in a $O(n)$ algorithm when $k \in O(n)$.

To be properly tested, every submission must compile correctly as submitted. If your submission does not compile for any reason (even trivial mistakes like typos), it will receive at most 5 out of 25. The best way to make sure your submission is correct is to download it from ConneX after submitting and test it. You are not permitted to revise your submission after the due date, and late submissions will not be accepted, so you should ensure that you have submitted the correct version of your code before the due date. ConneX will allow you to change your submission before the due date if you notice a mistake. After submitting your assignment, ConneX will automatically send you a confirmation email. If you do not receive such an email, your submission was not received. If you have problems with the submission process, send an email to the instructor **before** the due date.