CS 341: Algorithms @ Waterloo **Last Updated:** Jan 11, 2016
Homework 1 **Due Date: Jan 25, 2016, 11:59pm**

There are totally 55 marks (including the bonus). The full mark is 50. This homework is counted 5% of the course grade.

Please write pseudocode for the algorithms, prove their correctness and analyze their time complexity.

---

1. **Programming Problem: Counting Inversions** (10 marks)

   Implement the $O(n \log n)$ algorithm for counting inversions. For a list of integers L, a pair of indices $i < j$ is said an inversion if $A[i] > A[j]$. Notice that it is possible that $A[i] = A[j]$, this is not counted as an inversion. For example, the list A = [9, 6, 9, 7] has 3 inversions. These inversions are (1, 2), (1, 4), and (3, 4).

   Input: Your program will read a list of n positive integers from the standard input. Each line contains a list of decimal integers that are divided with whitespace (e.g. 13 15 6). If there are multiple lines in the input, the lists should be concatenated sequentially. Leading and trailing spaces in each line should be ignored. Empty or space-only lines should be ignored. Other than these, you can assume the test cases are in the correct format. You can assume each integer is $< 2^{31}$ and $n < 2^{31}$.

   Output: Your program will output a single number as a separate line to the standard output. This number should be the number of inversions of the input list. Use a 64-bit integer to avoid overflow of the precision.

   What to submit: Your C++ program and the make file. The detailed submission instruction will be released before the due date.

2. **Written Problem: Solving recurrences** (10 marks)

   Solve the following recurrence relations:

   (a) $T(n) = T(2n/3) + T(n/3) + n^2$.
   (b) $T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n$.

3. **Written Problem: Incomparable Pairs** (10 marks)

   Given n points on the 2D-plane $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, design an algorithm to count the number of *incomparable pairs*. A pair of points $(x_i, y_i), (x_j, y_j)$ is comparable if $x_i \geq x_j$ and $y_i \geq y_j$, or $x_i \leq x_j$ and $y_i \leq y_j$; otherwise they are incomparable.

   You will get full marks if the time complexity is $O(n \log n)$ and the proofs are correct. (Partial marks will be given to correct solutions with time complexity $o(n^2)$.)

4. **Written Problem: Finding Maximum Space**

This question has two parts.

(a) (10 marks) Imagine an advertising company would like to post a huge poster in Toronto harbour front. There are n consecutive bulildings there, with positive height $h_1, \ldots, h_n$ and unit width as shown in the figure. Design an algorithm to find the maximum rectangular space to post their poster. Stated mathematically, find $1 \le i \le j \le n$ to maximize $(j - i + 1) \cdot \min_{i \le k \le j}\{h_k\}$.

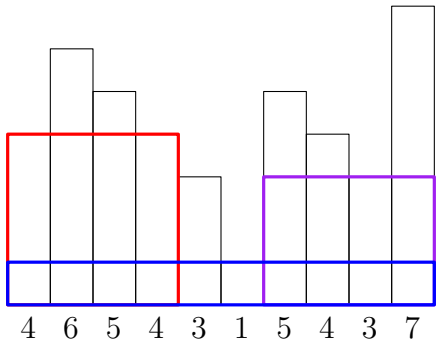

Figure 1: These are three possible solutions. The largest one has area $4 \times 4 = 16$.

You will get full marks if the time complexity is $O(n \log n)$ and the proofs are correct.

Bonus: There are up to five bonus marks if an algorithm with time complexity $O(n)$ is provided with correct proofs.

(b) (10 marks) Imagine the government would like to build a huge park in the city. The city is an $n \times n$ grid, with some units occupied. Use (a) or otherwise, design an algorithm to find the maximum (axis-aligned) rectanglular unoccupied space to build the park.

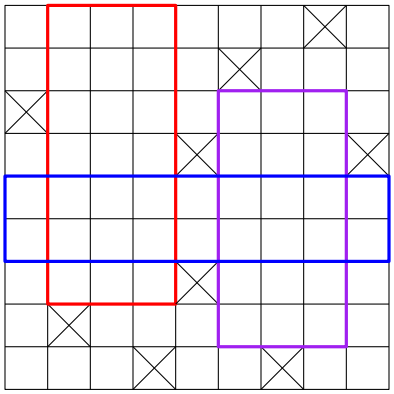

Figure 2: These are three possible solutions. The largest one has area $7 \times 3 = 21$.

You will get full marks if the time complexity is $O(n^2)$ and the proofs are correct. You can assume that there is an $O(n)$ time algorithm for part (a).