For these problems, assume that you are using Insertion Sort with a sentinel (as done in class). Assume that the first and second smallest elements are next to each other (somewhere in the input), the third and fourth smallest elements are next to each other, the fifth and sixth smallest elements are next to each other, etc. Here is a GENERAL EXAMPLE:

50, 60, 40, 30, 90, 100, 10, 20, 80, 70

(The algorithm does not know this, and executes without the extra information. Assume that the problem size n is even.) For each problem show your work.

Problem 1. Assume that each pair is in reverse order (so that the second smallest element comes before the smallest, the fourth smallest element comes before the third smallest, etc.). For example,

60, 50, 40, 30, 100, 90, 20, 10, 80, 70

What is the exact number of comparisons in the best case **as a function of** n?

Problem 2. Assume that each pair is in order (so that the smallest element comes before the second smallest, the third smallest element comes before the fourth smallest, etc.). For example,

50, 60, 30, 40, 90, 100, 10, 20, 70, 80

What is the exact number of comparisons in the worst case **as a function of** n?

Problem 3. Assume that each pair is in a random order, and that the pairs are in random order. The GENERAL EXAMPLE is a possible order. What is the exact number of comparisons in the average case **as a function of** n?

Problem 4.

(a) Now assume that you do know that the first and second smallest elements are next to each other (somewhere in the input), the third and fourth smallest elements are next to each other, the fifth and sixth smallest elements are next to each other, etc. (as in the GENERAL EXAMPLE). Give an efficient algorithm to sort such a list based on insertion sort with a sentinel. Write the pseudo-code.

(b) Analyze the exact number of comparions in the worst case **as a function of** n.