Problem 1. Assume that you are sorting an array with an even number of elements using Bubble Sort. 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. For example, assume $n = 10$, the input might look like

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

(The algorithm does not know this, and executes without this extra information.) For each part show your work. Your answer for each part should be a function of n.

(a) Assume each of the above pairs are out of 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 exchanges in the best case?

(b) Assume each of the above pairs are 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 exchanges in the worst case?

(c) Assume each of the above pairs are in a random order. What is the exact number of exchanges in the average case?

Problem 2. We are going to repeat Parts (a) and (b) of Problem 1, but with groups of arbitrary constant size k (rather than size 2). Assume that you are sorting an array using Bubble Sort, where $k|n$. Assume that the smallest k elements are grouped together (somewhere in the input), the next smallest k elements are grouped together (somewhere in the input), etc. Your answer for each part should be a function of n and k.

(a) Assume each of the above groups are in reverse order (which means that the elements within a group are reverse sorted). What is the exact number of exchanges in the best case?

(b) Assume each of the above groups are in order (which means that the elements within a group are sorted). What is the exact number of exchanges in the worst case?

Problem 3. Formulate Bubble Sort as a recursive method. That is, given an array `A` of length n consisting of numbers, write a method `BubbleR` in pseudo-code that processes `A` in some way and then calls `BubbleR` with an array of smaller length. You can assume that when it finishes running, the sorted data is in `A`.