COP 3502 Suggested Program Edits: Brute Force, Sorting (Week 7 Programs)

1) Edit max jumps so that instead of taking calculating the max jumps in a permutation of 0, 1, 2, …, n-1, that the input is a set of n positive integers in between 1 and 100. So, for example, if the input is 12, 9, 6 and 22, then the program should discover that the sequence 6, 22, 9, 12 has a jumping distance of 32 (|6 - 22| + |22 - 9| + |9 - 12| = 16 + 13 + 3 = 32.), which is the maximum possible of any permutation of 12, 9, 6 and 22.

2) Edit the derangements program so that for each slot i, the value stored there has to differ from i by at least 2. For example, for n = 4, we could have 2, 3, 0, 1, since |0 - 2| = 2, |1 - 3| = 2, |2 - 0| = 2 and |3 - 1| = 2. This should result in fewer permutations being printed than the regular derangement program.

3) In selection sort the way it was taught in class, we place the maximum element of the array at the end first. Reverse the logic so that the first item that is selected is the smallest, the second item selected is the second smallest, etc.

4) Rewrite merge sort so that it splits the array into the left size that is one third (roughly) the size of the full array and the right side so that it is two thirds the size of the fully array. Run this version against the usual version and time the code on large arrays. Is this version faster or slower? Why?

5) Edit the posted file with quick sort and try a base case of various sizes ranging from 5 to 100, where in the base case, you sort the subarray using insertion sort. Time the various versions on large arrays. Which one works the best?