1. (9 points) One advantage we discussed for bubblesort compared to the others is that it stops quickly if the array is already sorted. In fact, this property is not unique to that algorithm.

   (a) Explain why insertion sort (as coded in the slides from lecture) is also faster than its worst-case asymptotic running time would suggest. What is it running time in this case?

   (b) Selection sort as coded in class will take $O(n^2)$ time even when the array is already sorted. Modify the code so that it completes in only $O(n)$ time on an already-sorted array. (Hint: You’ll need to keep some extra information as you seek the smallest value.)

   (c) Repeat the previous part with mergesort. (Again, you’ll need to do extra work before making the recursive calls to test if the array is already sorted.)

2. (5 points) In lab, you saw the basics of creating a Java GUI layout, with layout managers such as `FlowLayout`, `GridLayout`, and `BorderLayout` applied to frames and panels nested inside each other. This same idea is applied in other GUI toolkits as well. Look at the GUI of a program that you use (Eclipse, Word, etc) and identify how you would create its layout using what you learned in lab. (There may be parts you cannot recreate since we only learned a couple of components and layout managers, but do the best that you can.) If the program you analyze is not available on Knox AnyWare, include a sketch or screenshot of it.