Searching and Sorting

I give my students a variation on this assignment where I ask them to turn in a writeup with their answers.

1. When measuring the worst case performance of sequential and binary search, we typically do so by counting the number of times we compare the key against data in the list. Why didn't we do something simpler, like measuring running time with a stopwatch?

2. In the case of quantifying how much work binary search takes, there's actually a subtle distinction between

   - counting "the number of times we see if the key matches an item in the list," and
   - counting "the number of times we compare our key against an item in the list."

   Explain precisely how the answer you would get for your worst-case count in the second case differs from the first case. What criteria would make sense to use in deciding which of these two counts would be better to use?

3. Here is a list of numbers:

   ```
   [12, 6, 3, 9, 10, 4]
   ```

   Show how selection sort would handle this list. Specifically, show what the list would look like each time the positions of two numbers are changed.

4. Redo the above problem for insertion sort.

5. Here is a list of numbers:

   ```
   [3, 1, 4, 0, 5, 9, 2, 6]
   ```

   Show on paper a schematic demonstrating how mergesort would sort this list.