1. Explain what you think the worst-case, big-Oh complexity and the best-case, big-Oh complexity of bubble sort is. Why do you think that?

   The wort-case time complexity for bubble sort is O(n^2) which means we will need to have go through every elements for two nested loops. The best-case time complexity is O(n). This is because the list is already sorted, and we only need to go through each element in the first loop.

2. Explain what you think the worst-case, big-Oh complexity and the best-case, big-Oh complexity of selection sort is. Why do you think that?

   The worst-case and the best-case time complexity is O(n^2) because we will need to find the minimum number in the sub-list for n times. And finding the minimum value in list will need to go through all the elements in the list. Therefore, even though the list is already sorted, we still need to go through them n^2 times.

3. Does selection sort require any additional storage (i.e. did you have to allocate any extra memory to perform the sort?) beyond the original array?

   No. We will swap the two elements in the original array.

4. Would the big-Oh complexity of any of these algorithms change if we used a linked list instead of an array?

   No, we still need to go through all the elements n times.

5. Explain what you think big-Oh complexity of sorting algorithm that is built into the c libraries is. Why do you think that?