Practice Problems For Topic 4: Searching & Sorting  
CISC 121, Fall 2014

Please do at least problems 1-6. Problems 7-9 are optional.

1. Start with the skeleton modules and see if you can fill in the bodies of the searching and sorting functions. If you get stuck on a function, peek at the completed version for a clue, then look away and see if you can finish. Then go back later and see if you can write the whole function without help. Repeat as many times as necessary.

2. The `seqSearch` function always finds the index of the first occurrence of the target in the list. Create a modified version that finds the index of the last occurrence.

3. The `binSearch` function will find the index of an occurrence of the target in the list, but it won't necessarily be the index of the first (or the last) occurrence. Create a modified version that returns the index of the first occurrence of the target.

4. Write a "trinary search" function which searches for a value in a sorted list by dividing the list into three parts at each step.

5. As a "finger exercise" to make sure you're familiar with the bubble sort algorithm, write a new version that moves from the end of the list towards the beginning while swapping pairs of elements.

6. Similar to the above, write a new version of selection sort that finds the maximum element in the list and moves it to the end, then repeats.

7. Bubble sort can sometimes end early. Repeat some of my testing with lists that are "almost sorted". For example, for a list of size n, start with `list(range(n))` and randomly choose just a few pairs of elements to swap. How does the performance of bubble sort improve?

8. Think about whether the performance of selection sort will change with the lists that are "almost sorted". Do some tests to confirm your answer.

9. There's a version of bubble sort in which the first pass moves up the list and the second moves down and so on. What would be a possible benefit of this strategy? Try implementing it and comparing its performance to the original.