Sorting - Problem Set

1. We know that generic Quicksort picks the first element of the array as the pivot point. For each of the following arrays, tell whether or not Quicksort would have its worst case, $O(n^2)$.
   a) [1,2,3,4,5,6,7,8,9,10]
   b) [10,9,8,7,6,5,4,3,2,1]
   c) [10,1,9,2,8,3,7,4,6,5]

2. Put the letters E,X,A,M,P,L,E in order using a) Insertion Sort, b) Selection Sort, and c) Bubblesort. Show the intermediate collections of letters at each step.

3. How many comparisons would be needed to sort an array containing 25 elements using bubble sort in the worst case? Best case?

4. Sort the array with Count-Sort: [3,0,2,2,1,3]
   a) What does the C array look like after the first loop?
   b) What does the C array look like after the second loop?
   c) Show how to fill in the answer array, B.

5. Which sort would you use? Justify your answers.
   a) Sort a bunch of batting averages. Batting averages are numbers between 0 and 1 kept to 3 digits. Most batting averages are between .150 and .300.
   b) Sort 45 million real numbers as quickly as possible.
   c) Sort 45 million real numbers, but I only need to know the smallest 10 million. After that, I'll called the rest "sorted enough".
   d) Sort books by Dewey decimal numbers. Dewey decimal classifiers have up to three digits before the decimal and up to three digits after the decimal. You may assume any given Dewey decimal number is equally likely as another.
   e) Sort student records based on the number of semester hours they have (0-150). It is important that the sort be stable. There are more new students (on the low end of semester hours) then veteran students.
   f) Sort 25 financial records based on the record's owner name. The financial records take up a lot of space so it is best if they don't get copied.

6. Suppose you have a row of n disks of 2 colors, n/2 light disks and n/2 dark disks. You want all the dark disks to the right-end of the row and the left ones to the left-end row. The only move you are allowed to make is to exchange 2 neighboring disks. Design an algorithm that solves the puzzle, and tell how many moves it takes (in big O notation) to do so.

7. There are *n* integers in the range 0 to *k*. You need to be able to quickly answer the question of how many of the numbers fall in the range of [*a..b*] for any given numbers *a & b*. In order to answer the range question in constant time, you will do some preprocessing of the array. Write pseudocode for the two algorithms: Preprocess should take time O(n+k) and RangeQuery should take time O(1).

Preprocess(A[1..n], numbers in range 1..k): // Should return nothing but rather store some data
// structures for later use.

RangeQuery(int a, int b): // May refer to any data structure saved from the Preprocess method.

8. Given two arrays S1 and S2 (each of size n), and a number x, describe an O(nlogn) algorithm for finding whether there exists a pair of elements, one from S1 and one from S2, that add up to x. Feel free to call, rather than rewrite, any algorithm we have studied.

Find_Sum(S1, S2, x): // S1 and S2 are arrays of size n, and x is a number.

9. You make think that all research on sorting is over and done. Well, that's not at all true. When Python was created, a new sort was created which is used by Python and, since it is more efficient than Quicksort, it was adopted by Java as well. Use the Internet to look up this sort, named timsort. Tell how it works ***in your own words***. DO NOT COPY VERBAGE FROM THE WEB.