Algorithm, Take Home Midterm, due: 2005/11/17

9.1-6
You are given a sequence of \( n \) elements to sort. The input sequence consists of \( n/k \) subsequences, each containing \( k \) elements. The elements in a given subsequence are all smaller than the elements in the succeeding subsequence and larger than the elements in the preceding subsequence. Thus, all that is needed to sort the whole sequence of length \( n \) is to sort the \( k \) elements in each of the \( n/k \) subsequences. Show an \( \Omega(n \lg k) \) lower bound on the number of comparisons needed to solve this variant of the sorting problem. (Hint: It is not rigorous to simply combine the lower bounds for the individual subsequences.)

9.3-4
Show how to sort \( n \) integers in the range 1 to \( n^2 \) in \( O(n) \) time.

9-2 Sorting in place in linear time
a. Suppose that we have an array of \( n \) data records to sort and that the key of each record has the value 0 or 1. Give a simple, linear-time algorithm for sorting the \( n \) data records in place. Use no storage of more than constant size in addition to the storage provided by the array.

b. Can your sort from part (a) be used to radix sort \( n \) records with \( b \)-bit keys in \( O(bn) \) time? Explain how or why not.

c. Suppose that the \( n \) records have keys in the range from 1 to \( k \). Show how to modify counting sort so that the records can be sorted in place in \( O(n + k) \) time. You may use \( O(k) \) storage outside the input array. (Hint: How would you do it for \( k = 3 \)?)