R-3.19 Give a pseudo-code description of the removeElement dictionary operation, assuming the dictionary is implemented by a skip-list structure.

C-4.16 Given a sequence S of *n* comparable elements, describe an efficient method for determining whether there are two equal elements in S. What is the running time of your method?

C-4.18 Modify Algorithm inPlaceQuickSort (Algorithm 4.17) to handle the general case efficiently when the input sequence, S, may have duplicate keys.

C-4-19 Let S be a sequence of n elements on which a total order relation is defined. An ***inversion*** in S is a pair of elements x and y such that x appears before y in S but x > y. Describe an algorithm running in O(n log n) time for determining the number of inversions in S. **Hint:** try to modify the merge-sort algorithm to solve this problem.

C-4.25 Bob has a set A of *n* nuts and a set B of *n* bolts, such that each nut in A has a unique matching bolt in B. Unfortunately, the nuts in A all look the same, and the bolts in B all look the same as well. The only kind of comparison that Bob can make is to take a nut-bolt pair (*a*,*b*), such that *a* is from A and *b* is from B, and test it to see if the threads are larger, smaller or a perfect match with the threads of *b*. Describe an efficient algorithm for Bob to match up all of his nuts and bolts. What is the running time of this algorithm, in terms of nut-bolt tests that Bob must make?