Problem 1. Making Quicksort into “Slowsort”

Problem 2. Some Searching, Some Sorting, and Some Sums
Suppose you are given as input \( n \) positive integers and a number \( k \). Our goal is to develop an algorithm to determine if there are any four of them, repetitions allowed, that sum to \( k \). As an example, if \( n = 7 \), the input numbers are 6, 1, 7, 12, 5, 2, 14 and \( k = 15 \), the answer should be YES because \( 6 + 5 + 2 + 2 = 15 \).

(a) First solve the simpler problem of determining if there are any two numbers in a list that sum to \( k \) (repetitions allowed). Describe, in words, an algorithm that will achieve this goal in \( O(n \log n) \) worst case time. Explain how your algorithm achieves this complexity bound.

(b) Now observe that the sum of four numbers is the sum of two pairs of numbers. Carefully explain how we can solve the original problem (i.e., whether a four number sum exists). For full credit, show that your algorithm runs in worst case time \( O(n^2 \log n) \).

Problem 3. Sorting Phone Numbers
The input to this problem consists of a list of 7-digit phone numbers written as simple integers (e.g. 5551212 represents the phone number 555-1212). No number appears in the input more than once but there is no other limit on the size of the input.

Write precise (preferable Java-like) pseudocode for a method that prints out the phone numbers (as integers) in the list in ascending order.

Your solution must not use more than 2MB of memory. You should explain why this is true of your solution. (Note: No cheating! It cannot use any external storage—hard disks tapes punched cards the network an infinite tape etc. Assume that as input you will get an Iterator<Integer>).