CSE 326 DATA STRUCTURES
HOMEWORK 6

Due: **Wednesday, Aug 1, 2007** at the beginning of class.

1. **Sorting without arrays**

   Implement insertion sort using a linked list as your input, instead of an array. Specifically, assume you have a structure `record = [key: integer, info: data pointer, next: record pointer]`. Your input comes as a pointer to the head node of a linked list of records. Design pseudocode to implement a recursive version of insertion sort:

   - On an empty list input, return the empty list.
   - On a non-empty list input, recursively sort the tail of the list, then insert the head of the list into the appropriate place within the entire list.
   - You should never create or destroy record structures. You may temporarily unlink them from the list, but your list should essentially always contain all elements. Like sorting an array, this is a destructive operation: the original order of the list is lost. (So, you should not create a new list by copying elements from the old one.)

   Give a recurrence for the runtime of your algorithm, and solve it.

2. **Mergesort**

   Continuing with lists we would like to implement recursive mergesort. There are two major steps: splitting a list into two equal (or almost equal) size lists and is merging two sorted lists. Again, in the whole process the key field is never copied and the number of records remains the same.

   (a) Design pseudocode that given a list returns a pair `[first: record pointer, second: record pointer]` where first and second point to two list of equal (or almost equal) size. Hint: if the list has zero, one, or two records, then it is easy to return the right pair of lists. Otherwise, recursively split the list, except for the first two records, then return the result of putting the first record at the front of the first list and the second record at the front of the second list.

   (b) Design pseudocode that given two sorted lists returns the result of merging them into one sorted list.

   (c) Use split and merge to define the pseudocode for list mergesort.

3. **Some searching, some sorting, and some sums**

   Suppose you are given as input n positive integers and a number k. Write an algorithm to determine if there are any four of them, repetitions allowed, that sum to k. Your algorithm should run in time $O(n^2 \log n)$. Partial credit will be given if your algorithm is correct but takes longer than $O(n^2 \log n)$. 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 $6 + 5 + 2 + 2 = 15$.

   Prove your algorithm runs in the claimed time.

   Hint #1: First solve the simpler problem that determines if there are any two numbers that sum to k.

   Hint #2: The sum of four numbers is the sum of two pairs of numbers.

4. **Extra Credit**

   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 a program that prints out the phone numbers in the list in ascending order.

   Your solution must not use more than 2MB of memory. It cannot use any external storage either — disks, tapes, punched cards, the network, etc.

   Extra-Extra Credit: Implement your solution to this problem in C, C++, Java, or some other similar language. (In C, a solution should take about 20–30 lines, counting boilerplate code.)