Homework 4 (Posted 19th February, Due during or before class 26th February)

Policy for Programming Assignment: Problem 2 has a programming assignment. I would like to decouple the design from the programming part. If you are not sure of the solution for Problem 2, you can wait till 28th February, 11:59 pm to submit the program. The solution for the design part will be posted 26th February. You have 3 more days for this program. You can certainly submit your program on or before 28th, but if you want to wait please email the T.A. indicating your intention. You must email before 26th 11:59 p.m. (email must have a subject WAIT). All other solutions including the design part of problem 2 are due during or before class 26th February. If your name is John Smith, then name your program as JohnSmith.c and email it to yjkim78@gradient.cis.upenn.edu.

Problem 1: 8 pts Design an algorithm for deletion in an AVL tree (lazy deletion not allowed). You have to maintain the AVL property after deletion.

Problem 2: 8 + 10 pts Consider strings of 0s and 1s, (e.g., 0010, 101). A string s_1 is less than string s_2 if the first elements in s_1 is less than that in s_2, or if the first elements are equal in both, but the second element in s_1 is less than that in s_2, or if the first two elements are equal n both, but the third element in s_1 is less than that in s_2 and so on. In general, s_1 is less than s_2 if the first k elements are equal in both, but the $k+1$th element of s_1 is less than that of s_2 for any $k = 1, 2, ...$ For example, 0101011 is less than 01011. Also, if s_1 has length k and s_2 has length l, $k < l$ and the first k elements of s_1 and s_2 are equal, then $s_1 < s_2$. For example, 0110 is less than 01100. You have k strings. Total length of all strings is n. Give an algorithm to sort the strings in $O(n)$. For example, if you have $s_1, s_2, \ldots s_4$, and $s_1 < s_3 < s_2 < s_4$, then your algorithm should output the strings in the following sequence, s_1, s_3, s_2, s_4. Implement your algorithm in C. You may assume distinct strings.

Problem 3: 8 pts Every element in a linked list consists of 2 fields, one is a string, and another is a hash value of the string. There are n elements. The size of each string is O(log n). The size of the hash value is a small constant. Design an algorithm to find an element in the list using the hash values. You may assume that at most a constant k ($k > 1$) number of elements hash into the same value. Analyze the complexity if you are not allowed to use the hash value in any way. What would your answer be for both cases if the size of a string is $O(n^2)$?

Problem 4: 6 pts Consider a hash function, $h(k) = k \mathrm{mod} m$, where $m = 2^p - 1$. The inputs are the strings. We convert a string to an integer as follows. Let the string be $a_{k-1}a_{k-2}\ldots a_0$, then the corresponding integer is $\sum_{i=0}^{k-1} 2^{ip}\mathrm{ASCII(a_i)}$. The claim is that any 2 strings which are permutation of each other hash into the same position. Do you agree or disagree? Justify your answer.