Qualifying Examination – Fall 1998
Algorithms

1. (20 points) Design an implementation of the Abstract Data Type "Set" such that the following operations can be performed in $O(\log n)$ time.

$member(x, S)$	Return True iff $x \in S$.
$insert(x, S)$	Insert x into set S.
$delete(k, S)$	Delete the k^{th} smallest element from set S.

2. (20 points) You are facing a high wall that stretches infinitely in both directions. There is a door in the wall, but you don't know how far away or in which direction. It is pitch dark, but you have a very dim lighted candle that will enable you to see the door when you are right next to it. Design an algorithm to find the door by walking at most $O(n)$ steps, where n is the number of steps that you would have taken if you knew where the door is and walked directly to it. What is the constant multiple in the big-O bound for your algorithm?

3. (60 points) A string-processing language allows the programmer to break a string into two peices with a cost proportional to the length of the string. Suppose that a programmer wants to break a string into many pieces. The order in which the breaks are made affects the total amount of time used. For example, suppose that we wish to break a 20 character string after characters 3, 8 and 10 (numbering the characters from left to right starting from 1). If the breaks are made from left-ro-right, the first break costs 20 units of time, the second costs 17 units of time and the third costs 12 units of time, for a total of 49 time units. If the breaks are made from right-to-left, the cost would be 20, 10 and 8 for the first, second and third breaks respectively, for a total of 38 time units.

   Our goal is to find the order in which the breaks should be made to minimize the total amount of time. Let n be the number of breaks. The input to the algorithm is the length of the string and the numbers of the characters after which to break. The output of the algorithm should be the order in which the breaks should be made.

   (a) Consider the following greedy strategy: start by cutting the string as close to the middle as possible, and then repeat the same thing recursively in each half. Show that this greedy strategy does not lead to the optimal solution by giving a counterexample. (20 points)

   (b) Consider the following greedy strategy: start by making (at most) two cuts to separate the largest substring, and then repeat the same thing until finished. In case of a tie, we favor a cut that is close to the middle. Ties that still exist can be broken arbitrarily. Show that this greedy strategy does not lead to the optimal solution by giving a counterexample. (20 points)

   (c) Design a dynamic programming algorithm to solve this problem in $O(n^3)$ time. (20 points)