Introduction to Algorithms	**Review Questions for Prelim 1**
CS 482 Spring 2000	

The first prelim is *February 24 at 7:30 pm.* If your last name starts with the letters A-P, you should go to Olin 155; if your last name starts with the letters Q-Z, you should go to Olin 255. The prelim will be closed-book, closed-notes.

The questions below are meant to help in studying for the prelim. The questions are not meant to be handed in, and solutions to them will not be handed out. Rather, we will discuss them at the prelim review session in class on Monday Februrary 21. You can also discuss them with us in office hours.

The coverage of the prelim is roughly the same as Problem Sets 1-3, and Chapters 1-4 of the book (excluding Section 2.3). All handouts up to this point in the course are available in Upson 303.

(1) Consider a town with n men and n women seeking to get married to one another. Each man has a preference list that ranks all the women, and each woman has a preference list that ranks all the men.

The set of all $2n$ people is divided into two categories: *good* people and *bad* people. Suppose that for some number k, $1 \leq k \leq n-1$, there are k good men and k good women; thus there are $n-k$ bad men and $n-k$ bad women.

Everyone would rather marry any good person than any bad person. Formally, each preference list has the property that it ranks each good person of the opposite gender higher than each bad person of the opposite gender: its first k entries are the good people (of the opposite gender) in some order, and its next $n-k$ are the bad people (of the opposite gender) in some order.

Show that there exists a stable matching in which every good man is married to a good woman.

(2) Let S' and S be sequences consisting of 0's and 1's. We say that S' is a *subsequence* of S if S' can be obtained from S by deleting some of the symbols in S. (In other words, the symbols of S' occur in order, but not necessarily contiguously, in S. For example, 00110 is a subsequence of 0110111100.)

Give an algorithm that takes two sequences of 0's and 1's — S' of length m and S of length n — and decides in time $O(m+n)$ whether S' is a subsequence of S.

(3) Suppose you're consulting for a small computation-intensive investment company, and they have the following type of problem that they want to solve over and over. A typical instance of the problem is: they're doing a simulation in which they look at n consecutive days of a given stock, at some point in the past. Let's number the days $i = 1, 2, \ldots, n$; for each day i, they have a price $p(i)$ per share for the stock on that day. (We'll assume for simplicity that the price was fixed during each day.) Suppose during this time period, they wanted to buy 1000 shares on some day, and sell all these shares on some (later) day. They want to know: when should they have bought and when should they have sold in order to have made as much money as possible? (If there was no way to make money during the n days, you should report this instead.)

Example: Suppose $n = 3$, $p(1) = 9$, $p(2) = 1$, $p(3) = 5$. Then you should return *"buy on* 2*, sell on* 3*"*; i.e. buying on day 2 and selling on day 3 means they would have made \$4 per share, the maximum possible for that period.

Clearly, there's a simple algorithm that takes time $O(n^2)$: try all possible pairs of buy/sell days and see which makes them the most money. Your investment friends were hoping for something a little better.

(a) Give a divide-and-conquer algorithm that finds the correct numbers i and j in time $O(n \log n)$.

(b) Give a dynamic programming algorithm that finds the correct numbers i and j in time $O(n)$.