Algorithms and Applications (15-499)
Assignment #4 Due Date: 15 Apr 03

This assignment is only worth one half as much as the previous assignments since you are working on a project.
Do two of the following three problems.

Problem 1: Knuth-Morris-Pratt (KMP)
Recall that in the preprocessing step of the KMP algorithm for string matching, we compute values $l(i)$ for a string S (see page 6 of the Computational Biology 6 notes). In the following we assume sequences are indexed starting at 1. $l(i)$ is defined to be 1 plus the length of the longest suffix of $S[2..i-1]$ that matches a prefix of S. We define $l(1) = 0$.
For example, for the string `abcadabd`, l is given by $(0, 1, 1, 1, 2, 1, 2, 3)$.
The following is incomplete code for computing $l(i)$. The variable i tracks the current position for which we are computing $l(i)$. j tracks the position of the character that we match against $i - 1$. Fill in the two missing lines.

```
l[1] = 0;
j = 0;
for (i = 2; i <= n; i++) {
   while ((j > 0) && (s[i-1] != s[j]))
      j = _________ ;
   j = j + 1;
   l[i] = _________ ;
}
```

Argue (briefly and cleanly) that the total number of iterations of the while loop across all iterations of the for loop is bounded by n. The runtime of the routine is hence $O(n)$.

Problem 2: Treaps
Define the size of a node in a binary tree to be the number of nodes in the subtree rooted at that node. In a specific n-node binary tree T, let s_1, $s_2 \ldots s_n$ be the sizes of the nodes. Prove that the probability that an n node treap is isomorphic to T (has the same shape) is $\Pi_i 1/s_i$.

Problem 3: Searching Web Documents
Lets say I'm doing a startup called Googlemai that wants to handle more sophisticate queries about the web than are handled by the standard search engines, perhaps for a charge. One of the types of queries I want to be able to handle is the following. Given a set S of web pages, find other pages that are similar to it in terms of a fixed linear combination of outgoing links, incoming links and words in the document. How would I set this up using a single SVD? (Hint: Finding the first k eigenvectors of an appropriate matrix should be sufficient.)