Problem 1

The missing lines are $j = l[j]$ and $l[i] = j$.

In the i-th iteration, we already know that $S[i-l[i-1] \cdots i-2]$ matches with $S[1 \cdots l[i-1]-1]$. We start with $j = l[i-1]$. If $S[j] = S[i-1]$, then we know that $l[i] = j+1$ because $S[i-j \cdots i-1] = S[1 \cdots j]$. Otherwise, we need to search for a smaller prefix that matches a suffix of $S[1 \cdots i-1]$. Now, the next smaller prefix that matches a suffix of $s[1 \cdots i-2]$ is $s[1 \cdots l[j]-1]$, because the latter matches a suffix of $s[1 \cdots j-1]$. Thus we set $j = l[j]$ and repeat the whole process. So, the missing statement in the while loop is $j = l[j]$, and the other missing statement should be $l[i] = j$.

In each iteration of the while loop, the value of j decreases by at least 1, because $l[j]$ is strictly less than j by definition. Moreover, in the entire algorithm, j gets incremented only n times - once in each iteration. Since the value of j never goes below 0 , this implies that the while loop is used at most n times across all iterations of the for loop.

Problem 2

A treap is created by assigning random priorities to nodes in the tree. Number the nodes of the binary tree in in-order. Then, in order for a corresponding treap to be isomorphic to the tree T, the random priorities should be selected such that the priority assigned to every node is the maximum among priorities of all nodes in the subtree rooted at it.

Now, we can prove the statement by induction on the depth of the tree. The base case of $n = 1$ is simple. The tree is the single node, and the corresponding treap is isomorphic to this node with probability 1.

Consider the inductive step. Let the number of nodes in the tree be s, and the number of nodes in the left and right subtrees be T_L and T_R respectively. Then, the corresponding treap is isomorphic to this tree if and only if the root gets maximum priority among all the s nodes, and the treaps corresponding to the left and right subtrees are isomorphic to the respective subtrees. The first happens with probability $\frac{1}{s}$. From the inductive hypothesis, we have that the other two happen with probability $\prod_{i \in T_L} \frac{1}{s_i}$ and $\prod_{i \in T_R} \frac{1}{s_i}$ respectively.

The three are independent events. Thus we get that the treap is isomorphic to T with probability $\frac{1}{s} \prod_{i \in T_L} \frac{1}{s_i} \prod_{i \in T_R} \frac{1}{s_i} = \prod_{i \in T} \frac{1}{s_i}$.

Problem 3

In order to support queries on incoming and outgoing links, as explained in class, we set up a document by document matrix, that contains the following entries. $L_{ij} = 1$ if document i points to document j, and 0 otherwise. Then, we set up a document by term matrix with entries $W_{ij} = 1$ if document i contains term j and 0 otherwise.

Now, the matrix M obtained by appending L, L^T and W is an m by $2m+n$ matrix that contains information about incoming and outgoing links and terms in documents. We can now perform a "truncated" SVD of this matrix by computing $M_k = U_k \Sigma_k V_k^T$, the closest matrix of rank k to M. The query is computed by simply constructing the corresponding matrix for documents in the set S.