1. Write a pseudocode algorithm for finding all occurrences of a pattern $P$ in a text $T$ using the suffix tree of $T$.

2. The reverse of a string $S[0..m)$ is the string $S^R = S[m-1]S[m-2]..S[0]$. Describe an algorithm for finding the longest factor $S$ of $T$ such that the reverse $S^R$ is a factor of $T$ too. The algorithm should work in linear time on a constant alphabet.

3. Hamming distance is the edit distance with substitution as the only allowed edit operation. Let $ed_H(A, B)$ denote the Hamming distance of two strings $A$ and $B$ of the same length.
   
   (a) Suppose we have preprocessed the strings $A$ and $B$ so that the longest common extension for any pair of suffixes can be computed in constant time. Show how the Hamming distance $ed_H(A, B)$ can be computed in $O(ed_H(A, B))$ time.
   
   (b) Design an $O(kn)$ worst case time algorithm for approximate string matching with Hamming distance.

4. What is the number of distinct factors in the string "abracadabra"?

5. Give a linear time algorithm for computing the matching statistics of $T$ with respect to $S$ from the generalized suffix array of $S$ and $T$ and the associated LCP array (without constructing the suffix tree).

6. Prove Lemma 5.9. Hint: Generalize Lemma 1.18(b) (Lecture 2, slide 28) from three strings to many strings.