1. Simulate the Boyer Moore algorithm to locate occurrences of pattern ABCAAB in text ABAAABCAABCAACAAB
Write clearly at each stage which rule ("Bad Character" or "Good Suffix") you are applying.

2. Use Ukkonen's algorithm to construct the suffix tree for the string ATTAINING
Use the suffix tree that you constructed to determine if the substring TAINT occurs within the input string. How many comparisons does it take to answer the query?

3. In class we discussed how to use the suffix tree of a text T to see if a given pattern P occurs in T. Describe how you can use the suffix tree to find the locations of all the occurrences of P in the indexed string. What is the time complexity of the query?

How can you solve this problem without building a suffix tree? Compare the total time complexity of each method. In which case is the suffix tree preferred?

4. How can you use the suffix tree to quickly find out if there are any substrings in the text of length $\geq g$ that occur $k$ or more times?

Describe an algorithm for this task and analyze its time complexity. Optimally, your algorithm should take linear time.

5. Given a set $E$ of $k$ strings, we want to find every string in $E$ that is a substring of some other string in $E$. Assuming that the total length of all the strings is $n$, give an O(n)-time algorithm to solve this problem that uses the suffix tree.

6. We discussed how it is possible to index T with a suffix tree to search for a given pattern P (see question 4). Suppose that you are allowed to preprocess P which is given in advance, and then you have to search T online (without constructing a suffix tree for T). Sketch an idea of how it is possible to use a suffix tree of P to perform pattern matching on the text T.

Hint: Use the suffix links.
Hint: See "matching statistics" in Gusfield text.