1. The **Longest Common Prefix** problem is defined as follows:
   **Preprocess:** $D = \{S_1, \ldots, S_n\}, S_i \in \Sigma^m$, that is $D$ is a set of $n$ strings, each of which is of length $m$.
   **Queries:** $LCP(i, j)$ returns $k$ if $S_i[1 \ldots k] = S_j[1 \ldots k]$ but $S_i[k + 1] \neq S_j[k + 1]$.
   One trivial algorithm is to do no preprocessing and then simply scan string to find the $LCP$ at query time. However, suppose you were building a system where queries were very frequent. Give an algorithm for this problem which answers queries in constant time. The faster the preprocessing the better.

2. A palindrome is a string that reads the same forwards and backwards, like “Able was I ere I saw Elba” or “Lonely Tylenol” (in this case if you ignore the spaces). Given a string, a palindrome of the string is a maximal substring that’s a palindrome, so that “bbaaabbbbaaaa” has palindrome “bbaabb” and “aaabbbbaaa” and “aaaaa”, etc.
   (a) Prove that any string has an at most linear number of palindromes.
   (b) Give an algorithm for finding the palindromes of a string.

3. Suppose you are given a string $s$ of length $n$. Describe an $O(n)$-time algorithm to find the longest string $t$ that occurs both forwards and backwards in $s$. Your algorithm must use suffix trees.
   For example: If $s = yabxqebaz$, your algorithm should return $t = abc$ or $t = cba$ because both $abc$ and its reverse $cba$ occur in $s$ and no longer such string exists.

4. Let $S$ be a string of length $n$. Give an $O(n)$-time algorithm to find the longest repeated substring of $S$ such that at least two copies of the substring do not overlap in $S$. 