1 **Keyword tree merging, 10 points total**
Come up with an efficient algorithm for merging two keyword trees, $T_1$ and $T_2$.

2 **Simple suffix tree, 10 points**
Consider the string $s = \text{ACACACCA}$.

   a. Draw the lookup table for $s$ with a window size of 3.

   b. Draw the suffix tree of $s$. Label all the edges and nodes appropriately. Show the suffix links for all internal nodes.

   c. Describe the mechanism by which you will identify all length three repeats in the string from i) lookup table and ii) suffix tree. List the repeats.

3 **Suffix trees, 20 points total**
Prove or disprove the following statement: The suffix tree of a string and the suffix tree of its reverse have the same number of internal nodes.

4 **Prefix tree, 20 points total**
The classic use of suffix trees is in pattern matching. Once the suffix tree of a string $T$ is built, the existence of pattern $P$ in the string can be answered in $O(|P|)$ time, making suffix trees ideal for searching short patterns (such as motifs) in long strings (such as genomes). Your colleague defines a new data structure called *prefix trees* as follows: The prefix tree of a string in the compacted tree of all prefixes of the string. Can pattern matching be done efficiently on prefix trees? Why or why not?

5 **Substrings, 20 points total**
Give a linear time algorithm to count the number of distinct substrings of a string $S$ of length $n$. 
6 Minimal unique substring, 20 points

A substring $\alpha$ of $S$ is called a minimal unique substring if it satisfies the following properties:

1. $\alpha$ occurs exactly once in $S$ (uniqueness).
2. All proper prefixes of $\alpha$ occur at least twice in $S$ (minimality).
3. The length of $\alpha$ is at least $l$, for some given $l$.

Give an efficient algorithm to come up with all minimal unique substrings of a string $S$. 