INSTRUCTIONS:

1. Submit your solution to Gradescope by the due time; no late submissions will be accepted.
2. Type your solution (except figures; figures can be hand-drawn and then scanned); no hand-written solutions will be accepted.

Problem 1 (10 points).

In the KMP algorithm we defined spm_i for pattern P as the length of the longest substring of P that ends at position i, $i > 1$, and matches a prefix of P and that $P[i+1] \neq P[spm_i + 1]$. This definition does not apply to the case $i = |P|$. How should this case be defined? Justify your answer.

Problem 2 (10 points).

Prove that a suffix tree for string s has $O(|s|)$ nodes and $O(|s|)$ edges.

Problem 3 (10 points).

Prove that (1) each node in a suffix trie has a suffix link; and (2) each node in a suffix tree has a suffix link.

Problem 4 (10 points).

Design an algorithm to count the number of distinct substrings of a given string s in $O(|s|)$ time.

Problem 5 (10 points).

Given a set S of strings, design an algorithm to find every string in S that is a substring of some other string in S; your algorithm should run in $O(M)$ time, where M is the total length of all strings is S.

Problem 6 (10 points).

Let T be a suffix tree for string s. Let $str(u)$ be the string represented by node u in T, i.e., the string spelled out when walking from the root of T to node u. A node in T is called left-aligned if the occurrences of $str(u)$ in s are always preceded by the same character. For example, if $s = ababacb$ then the node representing ba is left-aligned since ba is always preceded by a, but the node with $str(u) = b$ is not left-aligned as sometimes b is preceded by a and sometimes by c. Give an algorithm runs in $O(|s|)$ time to find all the left-aligned nodes in T.

Problem 7 (16 points).

Let s and t be two strings; design an algorithm runs in $O(|s| + |t|)$ time to find the longest suffix of s that exactly matches a prefix of t.

1. Design such an algorithm using Z values (introduced in the Z-algorithm).
2. Design such an algorithm using suffix trie / suffix tree / generalized suffix tree.