Due: Monday 10/22 before 9:15

1. **(6.14)**
2. **(6.19)**
3. **(6.24)**
4. **[Implementation Question]** You are given a set of strings S_1, $S_2, \dots, S_m$. Each string's letters come from the same n-letter alphabet $A = \{a_1, \dots, a_n\}$. Furthermore, no letter is repeated within the same string, that is, for each string S_i, no letter appears more than once within S_i. Your goal is to find a hidden order on the letters, that is to sort them. You cannot compare the letters directly; the only way to find out that, for example, a_i should come before a_j is to find a string S_k in which a_i appears before a_j. In other words, the strings give you information about the hidden order $a_{\sigma(1)}, a_{\sigma(2)}, \dots a_{\sigma(n)}$, which is an extension of the *partial orders* specified by the strings $S_1, \dots S_m$.

   Design an algorithm to find an order compatible with all the strings, if one exists. Write down your algorithm and explain why it works. Then implement your algorithm. Your program should have a function `generalize` that takes as input a list of strings over the alphabet consisting of capital letters `A`, `B`, ..., `Z`, and returns an order compatible with all the given strings, again in the form of a string. If there is no such order, function `generalize` should return the empty string.

   Hint: build a directed acyclic graph.