1. (Left-over assignment 6 from the last week)
   Explain why Theorem 2.2.4 holds, and apply it to design a linear-time algorithm
   to accumulate the $l(i)$ values (or $l'(i)$ using the notation of the textbook) in linear
   time. (Assume that the $N_j(P)$ values have been computed.)

2. The extended version of the Boyer-Moore bad character rule is as follows: When
   a mismatch occurs between characters $P[i]$ and $T[h] = x$, shift $P$ to the right
   so that the closest occurrence of $x$ to the left of $i$ in $P$ gets aligned with $T[h]$  
   (Gusfield, page 18). The extended bad character rule can be implemented using
   either $\Theta(|\Sigma|n)$ or $\Theta(n)$ auxiliary space. Describe the structures and algorithms
   for preprocessing and looking up the shift values, and discuss trade-offs of the
   alternative implementations.

3. Draw an Aho-Corasick automaton for the set of patterns
   \[ P = \{ AT, AC, CAT, CATGAT, GACTAC \} \],
   indicating values of the goto, fail and output links.

4. Describe a simple record structure for implementing nodes of an Aho-Corasick
   automaton. How much space would a single node require? (The standard imple-
   mentation of a pointer takes 4 bytes.) How much space would be needed for an
   Aho-Corasick automaton built from all DNA patterns of length 10?

5. (Gusfield, Ex. 3.19) Show how to modify the wild-card matching method by re-
   placing array $C$ (which is of length $m > n$) by a list (or an array) of length $n$,
   While keeping the same running time.

6. (a) Present an algorithm for precomputing the occurrence masks $U$ of the Shift-
   And method for a given pattern. Assume that the alphabet $\Sigma$ is fixed, and
   use C-like bit operations.
   (b) Explain an extension of the Shift-And method to handle wild-cards efficiently
       (both in the pattern and in the text).