**NFA pattern matching**

Pattern-search programs take two inputs: a pattern given by the user and a file of text. The program determines whether the text file contains a match to the pattern, typically using some variation on NFA/DFA technology. Fully developed programs, such as grep, accept patterns containing regular-expression operators (e.g. union) and other convenient shorthands. Our patterns will be much simpler.

Let's fix an alphabet $\Sigma$ = {a, b, ....z, 0, 1, ....9}. Let $\Gamma = \Sigma \cup \{?, \#, +\}$. A pattern will be any string in $\Gamma^*$. A string *w* matches a pattern *p* if you can line up the characters in the two strings such that:

- When *p* contains a character from $\Sigma$, it must be paired with an identical character in *w*.
- The character ? in *p* can match any substring *x* in *w*, where *x* contains at least one character.
- The character # in *p* can match any numeric digit 0, . . ., 9.
- The character + in *p* can match exactly one character.

For example, the pattern "fleck" matches only the string "fleck". The pattern *p* = 2#3 will match strings such as "283", "203", but not "173" and not "2893". The pattern *p* = 273?f leck? will match a string *w* which starts with "273" immediately followed by some text of length at least one, followed by "fleck", followed by at least one more character (e.g. "273ab7sfleck8"). The pattern *p* = cs+27+ will match a string *w* which starts with "cs" immediately followed 1 character, then "27", then at least one more character (e.g., "csa273", "cs8275").
A text file *t* contains a match to a pattern *p* if *t* contains some substring *w* such that *w* matches *p*.

1. Design an algorithm which converts a pattern *p* to an NFA Np that searches for matches to p. That is, the NFA Np will read an input text file *t* and accept *t* if and only if *t* contains a match to *p*. Np searches for only one fixed pattern *p*. However you must describe a general method of constructing Np from any input pattern *p*.

2. Find several applications for pattern matching like this.