CS 341 Homework 5
Regular Expressions in UNIX

Regular expressions are all over the place in UNIX, including the programs grep, sed, and vi. There's a regular expression pattern matcher built into the programming language perl. There's also one built into the majordomo maillist program, to be used as a way to filter email messages. So it's easy to see that people have found regular expressions extremely useful. Each of the programs that uses the basic idea offers its own definition of what a regular expression is. Some of them are more powerful than others. The definition in perl is shown on the reverse of this page.

1. Write perl regular expressions to do the following things. If you have easy access to a perl interpreter, you might even want to run them.

(a) match occurrences of your phone number
(b) match occurrences of any phone number
(c) match occurrences of any phone number that occurs more than once in a string
(d) match occurrences of any email address that occurs more than once in a string
(e) match the Subject field of any mail message from yourself
(f) match any email messages where the address of the sender occurs in the body of the message

2. Examine the constructs in the perl regular expression definition closely. Compare them to the much more limited definition we are using. Some of them can easily be described in terms of the primitive capabilities we have. In other words, they don't offer additional power, just additional convenience. Some of them, though, are genuinely more powerful, in the sense that they enable you to define languages that aren't regular (i.e., they cannot be recognized with Finite State Machines). Which of the perl constructs actually add power to the system? What is it about them that makes them more powerful?

Regular Expressions in perl

`.`	Matches any character except newline
`[a-z0-9]`	Matches any single character of set
`[^a-z0-9]`	Matches any single character *not* in set
`\d`	Matches a digit, same as [0–9]
`\D`	Matches a non-digit, same as [^0–9]
`\w`	Matches an alphanumeric (word) character [a–zA–Z0–9_]
`\W`	Matches a non-word character [^a–zA–Z0–9_]
`\s`	Matches a whitespace char (space, tab, newline...)
`\S`	Matches a non-whitespace character
`\n`	Matches a newline
`\r`	Matches a return
`\t`	Matches a tab
`\f`	Matches a formfeed
`\b`	Matches a backspace (inside [ ] only)
`\0`	Matches a null character
`\000`	Also matches a null character because...
`\`*nnn*	Matches an ASCII character of that octal value
`\x`*nn*	Matches an ASCII character of that hexadecimal value
`\c`*X*	Matches an ASCII control character
`\`*metachar*	Matches the character itself (|, \., *...)
`(abc)`	Remembers the match for later backreferences
`\1`	Matches whatever first of parens matched
`\2`	Matches whatever second set of parens matched
`\3`	and so on...
x`?`	Matches 0 or 1 x's, where *x* is any of above
x`*`	Matches 0 or more *x*'s
x`+`	Matches 1 or more *x*'s
x`{`*m*`,`*n*`}`	Matches at least *m* *x*'s but no more than *n*
`abc`	Matches all of a, b, and c in order
`fee|fie|foe`	Matches one of fee, fie, or foe
`\b`	Matches a word boundary (outside [ ] only)
`\B`	Matches a non-word boundary
`^`	Anchors match to the beginning of a line or string
`$`	Anchors match to the end of a line or string

from Programming in Perl, Larry Wall and Randall L. Scwartz, O'Reilly & Associates, 1990.