Section 1: Regular Expressions

Regular Expression Reference Table

Symbol	Meaning	Example(s)
a	Literal character	a, any symbol in your alphabet Σ
ab	Concatenation of regex's (or literal characters) a and b	ab, (*abc*)(*def*)
ε	Empty string	ε
a | b	a or b	a, b
a*	0 or more a's	$\varepsilon, a, aaaa, aaaaa$
a+	1 or more a's: aa*	$a, aaa, aaaaa$
a?	0 or 1 a's: (a|ε)	ε, a
[*a-z*]	1 character in range *a-z*: ($a \mid b \mid \ldots \mid z$)	*a, b, c, d, e, f, …, z*
[*skj*]	1 of characters in bracket: ($s \mid k \mid j$)	s, k, j

1) Describe in English the set of strings generated by each of the following regular expressions and give two different strings it can produce:

i) (1 | 0)* 0

Non-empty strings of binary digits ending with 0

0, 10, 111100000, 01010

ii) ([A-Z][a-z]* | [0-9]+)

Sequence of letters starting with an upper-case A-Z followed by 0 or more lower-case letters, or non-empty sequence of decimal digits

A, Aa, Abczzz, 0, 3, 42, 17

iii) (ε | 4?0+1* X 3+)

Empty string or sequence of letters and digits made from the characters 0, 1, 4, X, and 3 that optionally begin with 4, contain at least one 0 followed by 1's followed by an X and ending in one or more 3's (i.e., it's hard to describe this collection of strings succinctly)

ε, 0X3, 401X333, 40000111X333

2) Write a regular expression to generate each of the following sets of strings:

i) All strings consisting of 0's and 1's (binary digits) with an even number of 0s

1* (0 1* 0 1*)*

ii) camelCase variable name in Java, where the alphabet is upper and lower-case letters with no digits or underscores

[a-z]+([A-Z][a-z]*)*

Note: this solution allows multiple upper-case letters to appear adjacent to each other. Challenge: produce a solution that does not allow adjacent upper-case letters.

iii) Non-empty strings of binary digits where each 1 directly follows a 0 (challenge: only use symbols in table up until *)
Challenge: (0 | 01) (0 | 01)*
Normal: (0+1?)+