#Name – Sahana
#Roll Number – BE17B038
#Time – 3-4 hours (consolidated)

ASSIGNMENT 2 – BT3051 – REGULAR EXPRESSIONS

1. Check if the following regular expressions match the target exactly from the given sample text.

(a)

Target: 18

Regex: [129]|18

The given regular expression will match for either a single digit which is 1/2/9 or the number 18. Since the target is number 18, the given Regex will match the Target **exactly**.

(b)

Target: There are two waiters, two footmen, two bellboys, and an Arab teenager, small, cheerful, and alert, who appears to be some kind of page. He is Zero.

Regex: There.*Zero

The given regular expression will match from the first word (There) to the last word (Zero) but however will not take the full stop (.) into account. Hence target is **not matched exactly** by the given regular expression.

2. Match/Mismatch

(a) \b\w{1,3}el{1,2}\b

The given word, (hence within word boundaries), should have only 1-3 letters before it encounters the letter 'e'. 'e' should then be followed by 1/2 occurrences of letter 'l'.

(b) \b(Q\w*)|([^c]{3}q\w{3})\b

The given word (hence within word boundaries), should either start with capital letter Q or should have 3 letter, which are not the letter 'c', and then followed by 'q' and 3 letters in order.

(c) \b\w*(((I|i)ni)|Iti)\w*\b

The word should either have Iti or I/I – ni. This can either be preceded or succeeded by alphabets.

(d) \b[0-9]{4,5}\b

The word should have 4 or 5 digits.

3. Regular expression examples

(a) \<a\w*e\w*i\w*o\w*u\w*\>

The word (hence the word boundary) should start with a, and have all the vowels in it with each vowel separated by either 0 or more letters in between.

Match: anemious, anteriour, aerious, affectious

Mismatch: placentious, materious, after, ambiguous

(b) \>\w*\w{2}\w*\w{2}\w*\>

The word (hence the word boundary) should atleast have 4 letters (alphabets) and at places with \w* - it might have 0 or more letters. Hence, any word with 4 or more letters will satisfy these criteria.

Match: tree, apple, mango, banana

Mismatch: pan, can, is, an

(c) ^(\d{3})\d*(\1\d){1,}\$

This is supposed to occur at the beginning of a line. The first group should contain a set of 3 digits. This group may or may not be followed by a series of digits. The second group must contain a repetition (exact)of the first group, followed by some digit. The second group must be present at least once. More repetitions are allowed. At the end of this tag, there must be a $ symbol.

Note – As long as there's a fresh start, this criterion is not fulfilled.

Match: 0900902$, 1231234$, 0001234500060007$, 98769876$

Mismatch: 123454321$, 111$, 123545674567$, 012312343$

(d) \w+\'\w+

Two set of letters (atleast 1 letter per set) are separated by an apostrophe (')

Match – Sahana's, it's, your's, don't

Mismatch – words, ape^s, 'sorted, 1234

(e) ^(((\d)\d)\d)\d\1\2\3\$

This is supposed to occur at the beginning of a line. First group consists of the first 3 digits, second group consists of first 2 digits, and the third group consists of the 1st digit. The target for this expression is 3 digits followed by 1 digit which are then followed by

elements of group 1, group 2 and group 3 in order. The target expression ends with a $ symbol.

Match – 1234123121$, 9876987989$, 1010101101$, 0000000000$

Mismatch - 1234123233$, 000000000$, 1234567890$, 123412312I$

4. Regular expressions

(a) \b([+-]|)[0-9]+([.][0-9]*|)\b

Explanation – The number can be either positive or negative, Also it is not necessary t mention + when the number is positive. Real numbers are represented with decimal points here. Hence there must be atleast 1 digit before the point, and either any number of digits after the point or no point at all (integers). The number is put in word boundaries to not match alphanumeric values.

Note – Alphanumeric expressions are equivalent to a single word with letters and digits in it. These expressions are avoided in the above regex.

(b) (0[1-9]|1[0-2])[/._-]([123][1]|[012][1-9]|[123]0)[/._-]([01][0-9][0-9][0-9]|20[01][0-9])

Explanation –

Month – There are only 12 months in a year and 00 does not exist. January to October are mentioned as 01-10.

Date – Every month has 31 days.

Year – This expression holds true for every year A.D. (0000-2019)

The numbers can be separated by either of the following: / . _ -

Note – The above expression does not consider leap year and hence February not having days 29/30/31. In fact, every month has 31 days.

(c) [^/]*\.\w{3}$

Explanation – Every character of the filename should not be a /. Filename is followed by a dot and a 3-letter extension.

(d) ATG([ATGC]{3})*((TAA)|(TAG)|(TGA))

Explanation – The longest ORF starts with ATG, followed by triplet sets of nucleotides (optional). And then matches the last stop codon in the same ORF.

(e) AUG([AUGC]{3})*?((UAA)|(UAG)|(UGA))

Explanation – The shortest ORF starts with AUG, followed by triplet sets of nucleotides (optional), however stops when the first stop codon is encountered.

(f) (?=.*?[A-Z])(?=.*?[a-z])(?=.*?\d)(?=.*?[^a-zA-Z0-9*@]).{7,15}

Explanation - ?= is Positive lookahead which Matches at a position where the pattern inside the lookahead can be matched. Hence it does not depend on the order of occurrence of the regex within that parenthesis. Hence, the above expression checks for 1 capital letter, 1 small letter, 1 digit and 1 special character, and the final length of the password should be between 7 – 15 characters only.