BT3051 Assignment 3
Roll Number: MM14B022
Collaborators: MM14B005
Time: 5:00

Question 1

(a) `ana$`
Any expression which **ends** with 'ana' matches. We can even use the regex 'a$'.

(b) `B.{7}`
Any expression starting with B should have a total length of 8 characters.

(c) `^\w*(\w)\1\w*$`
Any expression which has a repeating character right next to each other. For example, 'rr' in Barrack. There should not be any character between repeating characters.

(d) `^\d{5}$|^\d{4}$`
Any expression made only of digits and should either contain 4 or 5 digits.

(e) `^[^y]+$`
Any word that doesn't contain letter 'y' in it.

Question 2

(a) The symbols `\<` and `\>` in grep application, match the empty string at the beginning and the end of a word respectively. The string should start with 'a'. It first checks for 'a' then it can any number of word characters in between, then it checks for e and any number of word characters and so on, it finally checks for u and any number of word characters after it in the end. So basically the string must contain a, e, i, o, u anywhere in the string but in the same order.

Match: aeiou, akebinolu, aeiou, aaeeiiioonuukapil

Mismatch: kaeiou, aeioa, ae3i2o1, aueoiy

(b) The symbols `\<` and `\>` in grep application, match the empty string at the beginning and the end of a word respectively. So the regex matches any word that has more than 4 letters because there are two `\w{2}`. Between those pair of letters there can be any number of letters or even no letter. Same is the case for before and after the pair of letters.

Match: kapil, anythinmorethanfour, `<onlyinsidewillmatch>`, Anyword-Matches
Note: In the case where angular brackets are there only the inside letters will match, the angular brackets won't be a part of the match.
Note: In the case of 'AnywordMatches' please note that there is no hyphen, it is Latex's automatic formatting.

Mismatch: `<>`, `<abc`, abc, 12a

(c) In this regex the first capturing group captures the three digit number and next it can have any number of digits. Inside the second capturing group contains three digits of first capturing group and one extra letter. Now regex inside second capturing group should appear more than once after that because of `{1,}` which means `{min,max}`.

Match: 1231236, 123412351235, 123412351230, 12312351239, 12345664512331238

Mismatch: 12312365, 12345678, 1231234123, 4565876523

(d) In this regex there needn't be '\' before " ' ". In Grep it works without the backlash. This regex matches those which a have atlease one word character before and after " ' "

Match: C0u1dn't, Sh0uldn't, needn't, ain't

Mismatch: Kapils', n0thing, 'nomean1ng, after'

(e) This regular expression has a capturing group inside a capturing group which is inside another capturing group. So the outermost capturing group becomes the first capturing group and as you go inside capturing groups become second and then third.

Match: 1234123121, 9876987989, 9999999999, 4560456454, 2577257252

Mismatch: 1234112123, 9874998987, 99999999999, 0000001111, 456456454

Question 3

(a) `^[9 8]\d{7}$`

(b) `^(AE|AM|BT|BS|CE|CH|ED|EE|EP|HS|PH|OE|NA|MM|CS)1[0246]B(?!000)\d\d\d$`

(c) `^<(\w+)>.*<\\\1>$`

(d) `(?![ZXUOJB])[A-Z]{}`

(e) `^(?!(\w)\w*\1$).*$`