Übungsblatt 4
Reguläre Ausdrücke

HTWG-Konstanz
Gesundheitsinformatik / Angewandte Informatik - WS24/25
Theoretische (Grundlagen der) Informatik

Prof. Dr. Renato Dambe

06/07.11.2024

Aufgabe 1
Prüfen Sie, ob die Wörter zu den regulären Ausdrücken passen oder nicht.

Regulärer Ausdruck	**Zu prüfende Wörter**
[a-zA-Z] ∗ [0-9]?[a-z]+	Hallo4u, aBcDefg, Yolo3, Hauptstraße15c, HTWG, Test123abc456
[A-F]?(ghi) + [J-Z]∗	AFghiJZ, FghighiXXX, ghi, FCghiMAX, ghighighi
[^abc](def){3}(A|B)	xdefdefdefB, adefdefA, xydefdefdefA, zdefdefdefAB
ab[cd]+e(fg)*[hij]?	abe, abccefgj, abcdefghi, abcde

.

Aufgabe 2

Gegeben ist der folgende reguläre Ausdruck: `a[bc]?\D[ef]*(gh|ij)(lm){1,3}[^nop][q-u]+v\w`
Geben Sie an, welche der Zeichenfolgen von dem regulären Ausdruck als korrekt akzeptiert werden / wird und welche nicht. Wenn der Ausdruck nicht akzeptiert wird, markieren Sie bitte das erste Zeichen, welches zum Nichterkennen führt

	Wort	**ja**	**nein**
1)	`accghlmlmlmpuvw`		X
2)	`axijlmlmuuv_`	X	
3)	`ac1fijlmaqrsvx`		X
4)	`acbeehglmlmlm^noprsvw`		X
5)	`abcefghlmmqrsvw`	X	
6)	`aeghlmxv7`		X

Aufgabe 3

Gegeben ist der folgende reguläre Ausdruck: `(abc|def)gh?[i-l]+m(no)*p[^qrs]{2,}t(uv)?\wxyz`
Geben Sie an, welche der Zeichenfolgen von dem regulären Ausdruck als korrekt akzeptiert werden / wird und welche nicht. Wenn der Ausdruck nicht akzeptiert wird, **markieren Sie bitte das erste Zeichen**, welches zum Nichterkennen führt

	Wort	**ja**	**nein**
1)	`defgmnopzztaxyz`		X
2)	`abcghllimnonopzzxtaxyz`		X
3)	`abcghjlkmpqstuvbxyz`		X
4)	`defgkmpabt_xyz`		X
5)	`defimnopzztuvaxyz`	X	
6)	`abcghjkmpfgtu_xyz`		X

Aufgabe 4
Formulieren Sie für die Datentypen integer, decimal, code, id, dateTime und oid aus dem FHIR-Standard einen Wert, der zu dem entsprechenden regulären Ausdruck passt und einen, der zwar ähnlich ist, aber nicht zum Ausdruck passt. (Reguläre Ausdrücke unter `http://hl7.org/fhir/datatypes.html`).

Aufgabe 5
Formulieren Sie einen regulären Ausdruck, welcher für eine Eingabe prüft, ob die Eingabe ein korrekter Hexadezimal-String ist. Ein solcher String beginnt immer mit der Zeichenfolge 0x. Darauf folgt eine beliebig lange Folge an Ziffern (0 - 9), wobei auch die ersten 6 Buchstaben des Alphabeths (A - F) verwendet werden können (Groß- oder Kleinschreibung soll egal sein). Die Ziffernfolge darf keine führenden Nullen enthalten. Die Zahl 0 können Sie dabei ignorieren.

Aufgabe 6
Schreiben Sie ein Java-Programm, mit welchem Sie prüfen können, ob Strings von Regulären Ausdrücken als korrekt erkannt werden. Das Programm soll nacheinander zwei Eingaben entgegen nehmen. Die erste Eingabe ist der reguläre Ausdruck. Die zweite Eingabe ist der zu prüfende String. Als Ausgabe gibt das System 'true' aus, wenn der String als korrekt erkannt wurde und 'false', wenn der String nicht als korrekt erkannt wurde.

Aufgabe 7
Mit dem eben erstellten Programm können Sie nun verschiedene Reguläre Ausdrücke erstellen und testen.

a) Formulieren Sie einen regulären Ausdruck, der eine Uhrzeit auf die Form xx:yy:zz.sss hin prüft.

b) Formulieren Sie einen regulären Ausdruck zur Prüfung der Eingabe von Blutgruppen auf Korrektheit (Apos, ABneg, 0neg,...).

c) Formulieren Sie einen regulären Ausdruck, um Medikamenten-Intervalle (Morgen-, Mittag- und Abenddosis) einzugeben (1-0-0, 2-1-0, 0-0-1, ...). Zehn und mehr Einheiten werden dabei nicht verordnet und sollen daher nicht berücksichtigt werden.

d) Formulieren Sie einen regulären Ausdruck, der prüft, dass Eingaben keine Umlaute enthalten.

4.

integer
passend: -4000
ähnlich: -04000

decimal
passend: 3.14
ähnlich: +3.14

code
passend: one two three
ähnlich: one two three

id
passend: AAA.BBB.CCC
ähnlich: AAA+BBB.CCC

dateTime
passend: 2024
ähnlich: 2024-11-07T24:00

oid
passend: urn:oid:2.1.2.3.4
ähnlich: urn:oid:3.1.2.3.4

5.
^0x([1-9A-Fa-f][0-9A-Fa-f]*)$

6.
```
import java.util.Scanner;
import java.util.regex.Pattern;
import java.util.regex.Matcher;

public class RegexChecker {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      String regex = scanner.nextLine();
      String input = scanner.nextLine();
      Pattern pattern = Pattern.compile(regex);
      Matcher matcher = pattern.matcher(input);
      System.out.println(matcher.matches());
   }
}
```

7.
a) (?:[01]\d|2[0-3]):[0-5]\d:[0-5]\d\.\d{3}
b) (A|B|AB|0)(pos|neg)
c) [0-9]{1}-[0-9]{1}-[0-9]{1}
d) [^äöüÄÖÜ]*

Aufgabe 8

Öffnen Sie die Webseite `https://www.regex101.com/#pcre`. Gehen Sie nun auf die Wikipedia-Seite von Alan Turing (`https://de.wikipedia.org/wiki/Alan_Turing`) und kopieren Sie den dort befindlichen Text (mit Strg+A und Strg+C) in das Textfeld der Regex-Seite (Strg+V).

a) Formulieren Sie einen regulären Ausdruck, der nach allen Jahreszahlen aus dem letzten Jahrtausend sucht (1000-1999).

b) Formulieren Sie einen regulären Ausdruck, der nach allen Quellenverweisen im Text sucht (Eine Zahl in Eckigen Klammern)

c) Formulieren Sie einen regulären Ausdruck, der nach ISBN Nummern sucht

d) Formulieren Sie einen regulären Ausdruck, der alle Wörter sucht, die mit einem Großbuchstaben beginnen, danach kein a, e, n, t oder r enthalten und nicht am Ende des Satzes stehen (denen also ein Whitespace folgt).

e) Formulieren Sie einen regulären Ausdruck, der nach Wörten sucht, die mindestens 10 Zeichen lang sind, und die mit einem n oder r enden.

a) `1\d{3}`

b) `\[\d+\]`

c) `\d{3}-\d{1,5}-\d{1,7}-\d{1,7}-\d{1}`

d) `[A-Z][^aenrt\s]*\s`

e) `\w{9,}([nr])`