LIS590DC1 Theory & Practice of Data Cleaning **Spring 2017**
Dr. Bertram Ludäscher February 14

Assignment 1 – Regular Expressions

Due: Feb 25, 2017

The purpose of this assignment is to develop some simple regular expressions that can be used to match (or extract and transform) common types of data, e.g., dates, email addresses, URLs, etc. While developing and testing your regular expressions, you might find the website http://regexr.com/ very useful[1]. Note that you should not only test that your expressions *match* all desired patterns, but do ***not*** *match* in cases, where they shouldn't (i.e., beware of "false positives").[2] Another general hint: Starting from the examples given below, first try to describe the desired pattern in English. Then develop a regular expression that matches (only) that pattern.

Autograding and submission details: We are experimenting with new auto-grading tools and will share how to use these and how to upload your solutions via the Moodle forum. For now, please put all regex solutions in a .TXT file, with one line per regex solution.

Problem 1 (IP Addresses) An Internet Protocol (IP) address is a sequence of 4 numbers (between 0 and 255), using a single dot '.' as a separator: e.g., `172.16.254.1` or `127.0.0.1` are valid IPs (more precisely IPv4) addresses.

(a) Write a regular expression that matches IP addresses. Here you **don't** have to check that the numbers are in the range of 0 to 255. Try to keep it simple!

(b) **Extra Credit:** develop a more precise regex that also makes sure that examples such as `302.207.451.22` are ***not*** matched.

Problem 2 (Matching ISO-8601 Dates) There is an international standard called ISO 8601 for date and time-related date. Develop regular expressions for the following:

(a) Simple standard dates, e.g., `2017-02-14` … and similar.

(b) Date **combined with time** but ***without*** **time zone designators**, e.g., `2007-02-14T14:30`

Problem 3 (Matching Non-Standard Dates) In the US, dates are often written in the form MM/DD/YYYY, e.g., `02/14/2017`. Another common variant is: `February 14, 2017`. In some European countries it is common to write this date as `14.2.2017` or `14.02.2017`. Develop regular expressions that match the following non-ISO conformant (but widespread) ways to write dates. Note that you do ***not*** have to verify that the date actually exists, e.g. February 30, 2017 is OK for this exercise.

(a) `02/14/2017` (and dates like it, i.e. of the form *Month/Day/Year*). Note that there may or may not be leading zeros (e.g., `2/14/2017` is OK).

(b) `14.02.2017` (and similar, i.e., of the form *Day.Month.Year*). Again, there may or may not be leading zeros (e.g., `14.2.2017` is OK).

(c) `February 14, 2017` (and similar). Here the regexp should allow only correct spellings of the months January through December, but *independent* of case: (e.g., `FEBRUARY` in uppercase or `february` are OK, but `FEBRUAR` (missing "Y") is not OK). Hint: find a way to make regex matching case ***insensitive***.

[1] If you come across another great regex website, please share your finding via the Moodle forum!
[2] For the regex super-fans, there is RegexGolf: https://xkcd.com/1313/ (see also: explain xkcd)

Problem 4 (Email Addresses)
Develop a regular expression that matches email addresses. For our purposes, an email address consists of a *simplified local-part*, followed by the *at-sign* '@', and a case insensitive *domain*, which has *parts* (consisting of letters and optionally digits or a dash '-') that are separated by a dot '.' and where the last part should have at least two letters. The *simplified local-part* (similar to but different from the one described here) consists of one or more *pieces*, separated by a dot '.', and where each piece consists of one or more alphanumeric characters, a dash '-', an underscore '_', a percent sign '%', or a plus '+'.

Examples:

- `me@example.com`
- `more.with+symbol@foo.tv`

Counterexamples:

- `john..doe@example.com` `// no consecutive dots`
- `john.@example.com` `// no trailing dot`
- `.john@example.com` `// no leading dot`
- `A@b@c@example.com` `// not more than one "@"`
- `me@some.c` `// need two letters at the end`

Problem 5 (Extra Credit)
Write a Python program or OpenRefine function that can match any non-standard date from Problem 3 and reformat it as an ISO-8601 date (as in Problem 2).