Regular Expressions Assignment

IT&C 210B

This assignment involves composing regular expressions that match email addresses, social security numbers, and acceptable usernames. You will use the Unix Command-line utilities grep and sed.

Ended here. Do the following:

* Convert instructions to MarkDown
* Create a sample text sheet
* Create an answer sheet (possibly offer both .txt and .md)

While the following are theoretical situations, they could easily happen in real life. Use your knowledge of regular expressions to find a command to solve these situations. Regular expressions by themselves are good to know, but combining them with Linux commands such as **grep** and **sed** or the Python RE module will make them much more valuable to you. *As a note: You may need to run grep with –E to use special characters (like +).* I suggest researching these tools, along with reading their man pages, to help you with this assignment. You may also look for online “cheat sheets” with commonly used regular expressions. Please do this assignment on your own, though you can share resources with one another. Check your answers in Linux (for the first two) or on a site like regex101.com (set to Python and with flags g=global and m=multiline). Upload your completed homework to LearningSuite (put your answers in red, and a screenshot of it working ONLINE(Use the tests we give you. NOTHING in the bad examples should be highlighted) and in the given test directory (if that is part of the assignment), below each question and submit as a .doc or .docx).

1. Your website has many pages that have multiple different emails on them. Write a command that will return the locations of these pages. The emails can end in .com, .org, .net or .edu and must contain a “@” symbol with at least one character before and one character after the “@” symbol. Remember to make this Linux command recursive as you will be searching through an entire web root. (Hint: You’ll likely want to use the | symbol).

**Test Set**:

These **should** match:

right1@ads.com

right2@asdf.net

right3@dsas.org

right4@dwe.edu

right5@lds.org

These should **not** match:

WRONG@

@WRONG.com

WRONG@WRONG

WRONG@WRONGcom

WRONG@WRONG.oops

WRONG.com

1. You have recently taken over a website for a government agency. The last webmaster was very irresponsible. Users were required to enter their social security numbers, and the last webmaster stored these in a file named info.txt in plain-text. Run a command using sed to replace these numbers with “X”s. Assume that in these files Social Security numbers were all written in one of the following formats: 123-45-6789 or 123456789. (Hint: Your final answer will be in this format: sed –i ‘s/regularexpression(s)/XXXXX/g info.txt - make sure you look up the –i, s, and /g functionality so you understand why they are used.)

**Test Set**:

These **should** match:

123-45-6789

123456789

These should **not** match:

1234567890

123-456789

123 45 6789

12345-6789

--123456789

1. On your site you have a page that allows a user to create a username. The username must start with a capital letter and be followed by only lowercase letters. It must end in exactly two numbers and the whole username must be at least 8 characters long. Write a regex pattern that will check this. Assume you are embedding the regex in Python code, though you don’t need to include any code. (Hint: the answer can start with ^[A-Z]{1} and continue on from there. Notice the special use of {} brackets)

**Test Set**:

These **should** match:

Aabcde90

Qqqqqq00

Rrrrrrrrrrrrrrr99

These should **not** match:

ZAabcde00

AAabcd00

ZZZZZZZZ54

aaaaaaaaaa78

Aaaaa10

aA23aa88

1. On your website, you have a form that users use to create usernames. This form is processed using Python’s RE module. Create a regular expression pattern (not all of the Python code, just the RE pattern) that will check a string so that it has at least 8 characters and contains a lower and upper case letter, as well as at least one number. The uppercase letter(s), lowercase letter(s), and number(s) can be in any order. Only letters, digits, and an underscore should be acceptable (e.g., not other special characters, spaces, etc.). (Hint: Use the “lookahead assertions” of regex – e.g., (?=regex). Note that several lookahead assertions can be placed by each other. Try searching Google for “python regex lookahead password uppercase lowercase number” or similar terms to find examples that you can modify to meet the needs of this problem. Also, make sure you start with a ^ so that you require the beginning of the match to start at the beginning of the password field.)

**Test Set**:

These **should** match:

Aa123456

aA1234567

123456aA

Aa1bBccc

123\_56Aa

Ab23aC45333455

These should **not** match:

12345678

1234567a

1234567A

aAbcdefg

aB12345

Ba12345

12345ab

123456!a

234333aA [note the space at the beginning]