CSE/ISE 337 Assignment 3 – Regular Expressions (Spring 2019)

Due Date: Monday, April 22nd, 2019, at 11:55 PM

Instructions:

(a) When writing programs, you **must** use the techniques that are described in the lectures. You may **not** use methods, modules, packages that were not covered in this course.

(b) Your Perl scripts must use regular expressions to do perform matching and substitution against strings.

(c) Start working on this assignment right away; you will find it very difficult to finish if you wait until the last day.

(d) Please put the pragma **`use strict;`** and **`use warnings;`** at the start of all your Perl programs.

(e) Several of the questions for this assignment assume a Unix environment. Your code **must** run on the allv Linux machines. Running only on your own computer will cost you up to 15 points.

1. **Simple Matches** (15 pts total)

(a) Does the pattern `p|\s|r` match any of the following strings: (2 pts)

```
1. dancing room
2. [A]
3. report
4. card
5. 'lirr train'
```

If so, highlight the part of the string matched. If not, then state "no match".

(b) Does the patter `[0-8]{3}` match any of the following strings: (2 pts)

```
1. cse337
2. class2017
3. 3-24-2017
4. 256.781
5. 905-381-5472
```

If so, highlight the part of the string matched. If not, then state "no match".

(c) Does the pattern `c.*` match any of the following strings: (2 pts)

1. `cc`
2. `rock`
3. `coca-coloa`
4. `<c#>`
5. `rubik's cube`

If so, highlight the part of the string matched. If not, then state "no match".

(d) Does the pattern `\bbook\B` match any of the following strings: (2 pts)

1. `bookstore`
2. `booking`
3. `textbooks`
4. `"returned books'`
5. `audiobook`

If so, highlight the part of the string matched. If not, then state "no match".

(e) Does the pattern `[^O]*o+` match any of the following strings: (2 pts)

1. `balloons`
2. `FOODY`
3. `bookstore`
4. `"Look"`
5. `PoolRoOM`

If so, highlight the part of the string matched. If not, then state "no match".

(f) Write a Perl script to verify your answers to the above parts of the question 1. Your script should do the following:

1. Take the regular expression pattern as a command line argument to the script.
2. Read each string to be matched as input from `STDIN`.
3. If there is a match, print the input string to `STDOUT` with the matched part of the string enclosed between angle brackets, < and >.
4. If there is no match, then print a `"no match"` message to `STDOUT`.
5. Your program should continue to read input strings and attempt to match them against the regex pattern until the user hits `ctrl-c` or `ctrl-d` to manually end the program.

Hint: Please look at "Regex variables in Perl" slide from Lecture 12 for assistance in writing this script.

2. **Given Patterns** (15 pts)

What would the following regular expressions match or do? For each expression, do the following:

1. Give a complete and precise description of the function of the regular expression. Give reasons for your answer.

2. Give two good example strings to illustrate your explanation. A 'good' explanation is one that is as general and different from other examples as possible.

(a) `/"([^"]*)"/` (2pts)

(b) `/[-+]?\d+(\.\d*)?F\b/` (2 pts)

(c) `/(\D{2,}).*\[\1\]/` (2 pts)

(d) `/^[0-9]+\/\d+([+\-*\/]\=|([+]{2}|[-]{2}));$/` (3 pts)

(e) `s/(#?)1|one/[\1January]/ig` (3 pts)

(f) `/((.*?)\d)\s\2/` (3 pts)

3. **Finding Patterns** (15 pts) + (Bonus 2 pts)

For each part of this question please do the following:

1. Create a regex pattern that can perform the match described.
2. Create 2 example strings that illustrate the correctness of your regex pattern.
3. Use the script you wrote for question `1.f`, and your example strings, to verify the correctness of your.

(a) Find a pattern that removes any white spaces from the start and end of a line. **Note:** For this part of the question modify the `1.f` script so that it prints the modified input string to STDOUT. (5 pts)

(b) Find a pattern that matches any line of input that has the same word (note it is word, not pattern) repeated two or more times in a row. Assume that consecutive words are separated by one or two spaces. (5 pts)

(c) Find a pattern that matches a line in which every word is surrounded by spaces. An empty line always matches. (5 pts)

(d) Find a pattern that matches a line in which every word is adjacent to a space. An empty line always matches. (Bonus 2 pts)

4. **Application to Perl Programming** (13 points)

(a) (6 pts) Write a Perl script to process an HTML file. Your script should do the following:

1. Accept a filename as a command-line argument to your script.
2. Replace all paragraphs enclosed in `<p>` and `</p>` tags with ones that only have `<br><br>` at the end.
3. Remove all occurrences of the pair of tags `<span>` and `</span>`.
4. Print the changed content of the file to `STDOUT`.

Given the following sample file, included as `a3q4a-input.html`, as input:

```
<p>trees of green, red roses too</p>
<p>I <span><span>se</span></span>e them bloom for me an you</p>
<p>(what <span><span>a</span></span> <span><span>wonderful</span> </span>world)</p>
<p></p>
```

You should produce the following output:

```
trees of green, red roses too<br><br>
I see them bloom for me and you<br><br>
(what a wonderful world)<br><br>
<br><br>
```

(b) (7 pts) Write a Perl script that processes the output of the `echo $PATH` shell command. This command displays the content of the shell environment variable, `$PATH`. This variable contains a list of directory names. When you try to execute a command at the command-line, the shell will look for the program file corresponding to that command in each of the directories listed in the `$PATH` variable. Your script should do the following:

1. Execute the echo `$PATH` command. To execute a shell command in your Perl script see the example for the `qx` quote operator on the "Quoting Operations" slide from Lecture 9
2. Use a regex to extract the name of each directory in the output of the `echo $PATH` command.
3. Print each directory name to `STDOUT`. Print one directory per line.

Note: You must use the `m//g` regular expression construct to extract all the directory names and store them in an array first. Here is an example to extract all integers in a string and store them in an array:

```
$s = "We had 3 clubs with 10 people each in 4 months.";
@arr = $s =~ /\d+/g;
```

Submission Format (2 pts for the right submission format):
Submit a "`a3_LastName_FirstName.zip`" file that includes the following:

-- A `README.txt` or `README.pdf` file. It should contain your name, ID, and CS machine used (Linux allv machines or Windows 10 lab machines). It should also contain the answers to all questions that require a written response. Please label each of your answers clearly with the question number and part for which they are a response.

-- For all the programs you write, each program should be in a separate .pl file. Please name each program file using its question number and part. For example, the script written to test regexes against strings for the last part of question 1 should be in a file named "`1f.pl`".

Total Points: 60

Submission Instructions:
Assignments will be turned in through Blackboard. Make sure that your submission is correctly formatted before you turn it in, and that submission occurs before the deadline.

You can only submit twice. Only click "Submit" when you are sure you have followed all the submission requirements.

Late submissions will not be accepted. The due date is **11:55 PM, Sunday, April 22.**