Cairo University
Faculty of Computers & Artificial Intelligence
Theory of Computations

Assignment 1 (Regex)

Instructions

- The assignment is submitted in groups of **3 students** from the **same lab** or the **same TA**
- The Deadline for submission is **on Saturday 23/3 at 11:59 pm**
- Submission will be on Google Classroom, No late submission, or through e-mail submission is allowed.
- Please submit one compressed folder with the **.java files**. The folder name should follow this structure: **ID1_ID2_ID3_GROUP.zip**
- In case of **Cheating,** you will get a **negative grade** whether you give the code to someone, take the code from someone/internet, or even send it to someone for any reason.

Requirements

- Solve all the 10 problems.
- For all problems write a **Java program** to solve them using regular expression by following the **expected input & output format for each problem**.
- The **Java program** will take **ONE input text file** containing the input values for each problem, the program must save the output of each problem in **ONE output text file** following the required format for each problem.

Input file:

Starts with the number of the problem, followed by the input values for the problem, then **"end"** to indicate the input for this problem is finished and to move to the next problem.

Output file:

Starts with the number of the problem, followed by the corresponding output values of the input for the problem, then **"x"** to indicate the output for this problem is finished and to move to the next problem.

Problem#1

Validate email addresses using regex.

→ Sample:

Content in input file:

```
1
ramesh@gmail.com
tom@yahoocom
34234sdfa#2345
end
```

Content in output file:

```
1
valid email
invalid email
invalid email
x
```

Problem#2

Validate phone numbers using a regular expression.

Phone numbers can be written in many formats.

Examples of the common ways of writing phone numbers:

- 1234567890
- 123-456-7890
- (123)-456-7890
- 123.456.7890
- 123 456 7890

→ Sample:

Content in input file:

```
2
1234567890
123-456-7890
(123)-456-7890
1230456 7890
123/456/7890
end
```

Content in output file:

```
2
valid phone number
valid phone number
valid phone number
invalid phone number
invalid phone number
x
```

Problem#3

Validate dates using a regular expression.

The dates should be in one of the following formats:

- YYYY/MM/DD (e.g., 2024/03/11)
- YYYY-MM-DD (e.g., 2024-03-11)
- DD/MM/YYYY (e.g., 11/03/2024)
- D/MM/YYYY (e.g., 1/03/2024)

- DD/M/YYYY (e.g., 11/3/2024)
- DD-MM-YYYY (e.g., 11-03-2024)
- D-MM-YYYY (e.g., 1-03-2024)
- DD-M-YYYY (e.g., 11-3-2024)

→ Sample:

Content in input file:

```
3
11/3/2024
1st-03-2024
1 10 123
end
```

Content in output file:

```
3
valid date
invalid date
invalid date
x
```

Problem#4

Write a regular expression to validate IP addresses.

A valid IP address:

- Must contain numbers between 0 and 255
- There must be exactly three dots separating the four numbers

→ Sample:

Content in input file:

```
4
192.168.1.1
192.168.1.256
end
```

Content in output file:

```
4
valid IP address
invalid IP address
x
```

Problem#5

Validate all C++ variables using a regular expression.

→ Sample:

Content in input file:

```
5
x
x1y2
_hello
1x
bad name
!name
end
```

Content in output file:

```
5
valid C++ variable name
valid C++ variable name
valid C++ variable name
invalid C++ variable name
invalid C++ variable name
invalid C++ variable name
x
```

Problem#6

**** Given the alphabet doesn't contain special characters such as (*^&#@!?....)**

Validate strings in which the letter b is never tripled using regex.

No string contains 3 consecutive b's (lower or upper case).

→ Sample:

Content in input file:

```
6
bbb
bBb
aabbvs10nkn0b
end
```

Content in output file:

```
6
invalid string, has 3 consecutive b's
invalid string, has 3 consecutive b's
valid string
x
```

Problem#7

**** Given the alphabet is only (a,b)**

Using regex, extract strings with an odd number of a's and an odd number of b's from the user's input.

You should print **the number of matched substrings** and **the substring with its start and end indices.**

→ Sample:

Content in input file:

```
7
aabb
aabaaaaabaa
end
```

Content in output file:

```
7
*aabb*
number of matched substrings: 1
matched substring: ab
start index: 1, end index: 3
*aabaaaaabaa*
number of matched substrings: 2
matched substring: aabaaaaa
start index: 0, end index: 8
matched substring: ba
start index: 8, end index: 10
x
```

Problem#8

Extract words whose length is a multiple of 3 from an input string using regex.
You should print **the number of matched words** and **the words with their start and end indices.**

→ Sample:

Content in input file:

```
8
The cat is cute
an apple
end
```

Content in output file:

```
8
*The cat is cute*
number of matched words: 2
matched word: The
start index: 0, end index: 3
matched word: cat
start index: 4, end index: 7
*an apple*
No word matches
x
```

Problem#9

Write a regular expression to extract URLs from a text file, given the path of the text file as input.
You should print **the number of URLs and each URL with the line number, and start and end indices.**
The program can accept more than one text file.

→ Sample:

Content in input file:

```
9
C:\problem9file.txt
end
```

Content in output file:

```
9
*problem9file.txt*
Number of URLs: 4
URL: https://www.example.com/
Line: 1
start index: 36, end index: 60
URL: https://blog.example.com
Line: 2
start index: 55, end index: 79
URL: https://twitter.com/example
Line: 3
start index: 35, end index: 62
URL: https://www.example.com/example1/example2
Line: 4
start index: 43, end index: 84
x
```

** Given the problem9file.txt

```
Welcome to our website! Visit us at https://www.example.com/ to learn more about our services.
For the latest news and updates, check out our blog at https://blog.example.com.
You can also follow us on Twitter: https://twitter.com/example.
Do not forget to check for more details at https://www.example.com/example1/example2.
If you have any questions, feel free to contact us at support@example.com.
Thank you for visiting!
```

Problem#10

Write a regular expression to validate mathematical expressions.

Mathematical expressions:

- May contain numbers, floating points, or variables
- May have multiple variables, numbers, and operators on both the left and right-hand side.
- The equation is from the 1st degree only (There is no power)
- Consider operators (+-/*) only
- The equation does not contain parentheses

→ Sample:

Content in input file:

```
10
3x-3y=90
3x*-3y/z=90+n
=9
end
```

Content in output file:

```
10
valid mathematical expression
valid mathematical expression
invalid mathematical expression
x
```
