**ASSIGNMENT2: Using Regular Expressions**

**Problem Statements:**

1. **Android Logcat Service Summary**

An Android Application generates log info to an application process called logcat. The log messages in logcat look something like the following:

2020-10-24 14:45:47.727 26712-26712/com.example.loggen I/Perf: Connecting to perf service.

2020-10-24 14:45:48.048 26712-26712/com.example.loggen W/.example.logge: create CommonDcsHelper

The format is:

***date******time******PID****-****TID****/****package******priority****/****tag****:* ***message***

date is in yyyy-mm-dd format

time is in hours:minutes:seconds format

Make a summary with the start time, end time, and duration of services that appear in the logcat output file.

When a service starts, the ***message*** part of the line would be in the following format:

Starting Service <ServiceName>

Similarly, when it ends, it would be:

Ending Service <ServiceName>

2020-10-24 14:45:48.515 26712-26712/com.example.loggen D/[TESTAPP]: Starting Service Abc

2020-10-24 14:45:49.516 26712-26883/com.example.loggen D/[TESTAPP]: Ending Service Abc

There could be many interleaving services that would start and end. A particular service only runs once. You won’t have a given service start or end multiple times.

Generate a summary in the following csv format:

ServiceName,StartTime,EndTime,Duration(in milliseconds)

Abc,2020-10-24 14:45:48.515,2020-10-24 14:45:49.516,1001

**Notes**:-

* Make sure that the output is sorted by the start time.
* The input filepath will be specified as command line param 1. sys.argv[1] would be the input filepath.
* The algorithm should be a single pass algorithm. You can process a line only once. Write an algorithm that takes input line by line instead of reading the entire input as a big string.

**Execution:-**

$python <file\_name>.py sample\_logcat.txt

**Sample Input File:- (The entire logfile is present in the drive folder)**

2020-10-24 22:18:57.687 20593-20593/com.example.loggen W/.example.logge: sendCommonDcs: end

2020-10-24 22:18:57.687 20593-20593/com.example.loggen W/.example.logge: Accessing hidden method Landroid/widget/TextView;->getTextDirectionHeuristic()Landroid/text/TextDirectionHeuristic; (light greylist, linking)

2020-10-24 22:18:57.792 20593-20593/com.example.loggen D/[TESTAPP]: Starting Service Abc

2020-10-24 22:18:57.793 20593-20593/com.example.loggen D/[TESTAPP]: Starting Service Def

2020-10-24 22:18:58.148 20593-20804/com.example.loggen D/OpenGLRenderer: Swap behavior 2

2020-10-24 22:18:58.226 20593-20593/com.example.loggen I/Choreographer: Skipped 4 frames! The application may be doing too much work on its main thread.

2020-10-24 22:18:58.793 20593-20801/com.example.loggen D/[TESTAPP]: Ending Service Abc

2020-10-24 22:18:59.794 20593-20801/com.example.loggen D/[TESTAPP]: Ending Service Def

**Sample Output:-**

Abc,2020-10-24 22:18:57.792,2020-10-24 22:18:58.793,1001

Def,2020-10-24 22:18:57.793,2020-10-24 22:18:59.794,2001

1. **XML File Processing**

**Sample Input File**

<?xml version='1.0'?>

<tag0>

<tag2>

<name> abc </name>

<tag3>

<tag9> def </tag9>

<tag5> ghi

<tag7> mno

</tag3>

<tag3>

<tag8> def </tag8>

<tag5> ghi </tag5>

<tag6> jkl

<tag7> mno

</tag3>

<tag3>

<tag4>???\*\*</tag4>

<tag8>+??\*\*</tag8>

</tag3>

</tag2>

</tag0>

**Operations -**

* 1. Number of occurrences of a particular tag of the format <tag\_name> (tag name - input)
  2. Number of unclosed tags
  3. Number of tags with values in the following pattern

(+(0 or 5)?(3's multiple)\*(2's multiple) or +(2)?(2's multiple)\*(3's multiple)) (Multiples start from 1)

Eg:

* ???\*\* - Correct
* ++????\*\*\*\*\*\* - Correct
* +??\*\* - Incorrect
* +++++???\*\* - Correct
  1. Number of occurrences of a particular tag with **at least** 3 tags in the immediate next level of nesting (tag name - input) (max depth - 4)

Eg:

* For tag - Tag3
* The sample input file has two occurrences of tag3 having at least three components in the nesting

Execution:

$python <script\_name> XML\_file.xml

STDIN:

a tag3

b

c

d tag3

Expected output:

3

4

1

2

1. **Shell Script Processing**

**Sample Input File**

#!/bin/bash

if [ -f /etc/bashrc ]; then

. /etc/bashrc

fi

alias a=’b’

alias b=’c’

alias c=’ls -l’

dummy\_function ()

{

if [ -f file ]; then

echo “1”

else

echo “2”

fi

}

Operations:

1. Generate a shell script after processing the input (shell script)

* Remove all the **‘if’** statements (if - fi)
* Remove redundant functions (empty function body) **and aliases**
* Remove comments, blank lines and lines containing only whitespaces

(The shebang line -> #!/bin/bash should **not** be removed)

* Output filename - <SRN>\_A2.sh
* Display the file after saving

Execution:

$python <script\_name> shell\_script.sh

Expected Output:

#!/bin/bash

alias a=’ls -l’