Ministry of Education, Culture, and Research of the Republic of Moldova
Technical University of Moldova
The Faculty of Computers, Informatics, and Microelectronics

REPORT

Laboratory work no.4
Regular expressions

Executed by: Cornievschi Bogdan
st. gr. FAF-222

Verified by:
univ. assist. Crețu Dumitru

Chişinău - 2024

Introduction

Regular expressions are a foundational tool in the field of computer science, known for their efficiency and versatility in pattern matching and string manipulation. This report outlines the process of generating strings that match a given regular expression pattern using the Python programming language. Our focus is to demonstrate the practical application of regular expressions in creating strings within defined constraints, thereby showcasing their utility in tasks such as data validation, parsing, and text analysis.

Methodology

A Python script was developed to randomly generate strings that conform to specific regular expressions. These expressions dictate the structure of the strings, encompassing a variety of symbols each with a designated function, such as indicating the occurrence of certain characters or the frequency of their repetition. To illustrate the process, a function was also created to explain the sequence of operations performed by the regular expression in a human-readable format.

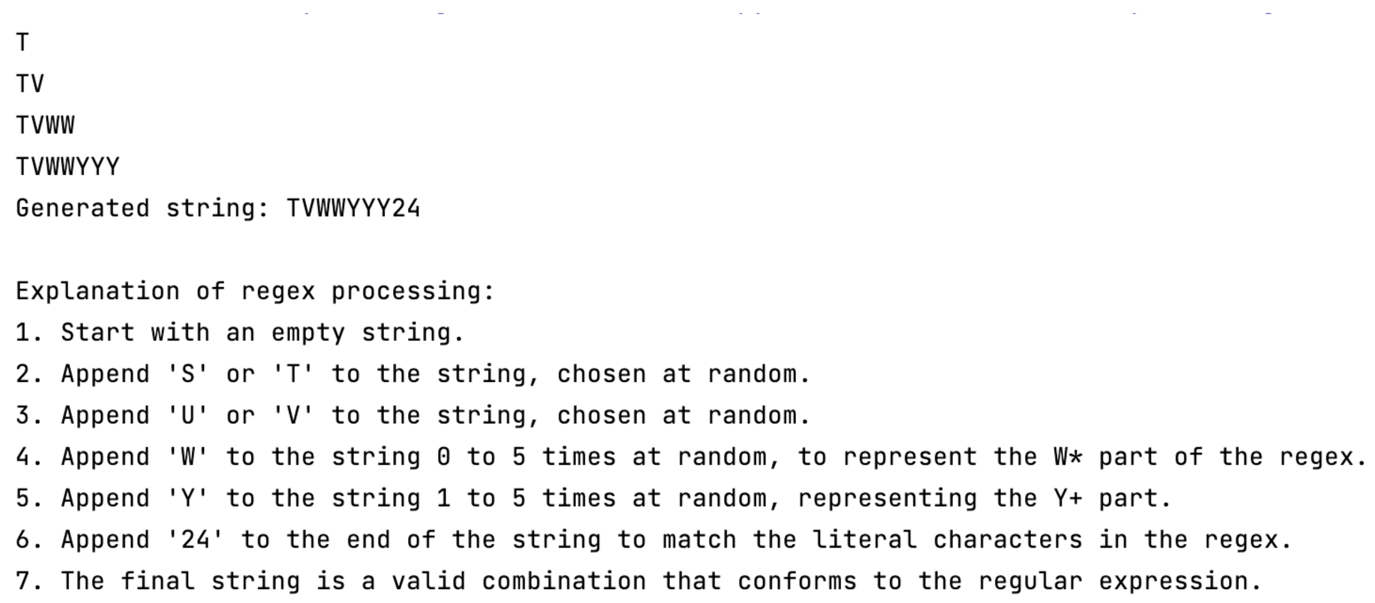
The regular expressions used for this exercise are complex, involving multiple operations such as grouping, selection, quantification, and specific character inclusion. The symbols within the expressions include parentheses for grouping ( ), the vertical bar for logical OR |, the asterisk for zero or more repetitions *, and the plus sign for one or more repetitions +. Literal characters are also part of the expressions, signifying exact matches required in the resulting strings.

Elements of the regular expressions

(S|T) - This means the expression will match either the character S or T
(U|V) - This means the expression will match either the character U or V
W* - This means the expression will match zero or more occurrences of the character W
Y+ - This means the expression will match one or more occurrences of the character Y
24 - Will be present in every expression

Output of the programs

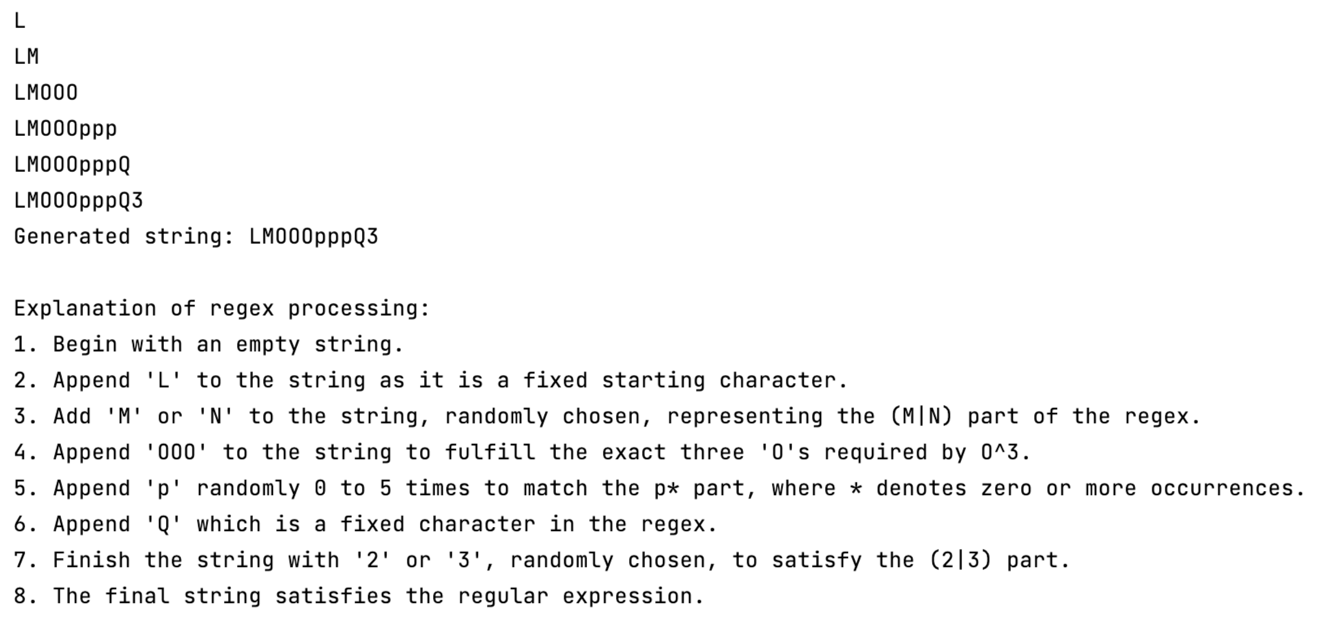
The first expression



```
T
TV
TVWW
TVWWYYY
Generated string: TVWWYYY24

Explanation of regex processing:
1. Start with an empty string.
2. Append 'S' or 'T' to the string, chosen at random.
3. Append 'U' or 'V' to the string, chosen at random.
4. Append 'W' to the string 0 to 5 times at random, to represent the W* part of the regex.
5. Append 'Y' to the string 1 to 5 times at random, representing the Y+ part.
6. Append '24' to the end of the string to match the literal characters in the regex.
7. The final string is a valid combination that conforms to the regular expression.
```

The second expression



```
L
LM
LMOOO
LMOOOppp
LMOOOpppQ
LMOOOpppQ3
Generated string: LMOOOpppQ3

Explanation of regex processing:
1. Begin with an empty string.
2. Append 'L' to the string as it is a fixed starting character.
3. Add 'M' or 'N' to the string, randomly chosen, representing the (M|N) part of the regex.
4. Append 'OOO' to the string to fulfill the exact three 'O's required by O^3.
5. Append 'p' randomly 0 to 5 times to match the p* part, where * denotes zero or more occurrences.
6. Append 'Q' which is a fixed character in the regex.
7. Finish the string with '2' or '3', randomly chosen, to satisfy the (2|3) part.
8. The final string satisfies the regular expression.
```

The third expression

```
R
RS
RST
RSTW
Generated string: RSTWXX

Explanation of regex processing:
1. Begin with an empty string.
2. Append 'R' 0 to 5 times, chosen randomly, to match the R* part of the regex.
3. Append 'S' as a fixed character.
4. Add one of 'T', 'U', or 'V' randomly, matching the (T|U|V) part.
5. Append 'W' as a fixed character.
6. Randomly choose one of 'X', 'Y', or 'Z' and append it twice, to match X^2, Y^2, or Z^2 if we assume ^2 means
7. The constructed string matches the regular expression pattern.
```

Conclusion

Throughout this laboratory exercise, we explored the practical application of regular expressions and the use of Python to generate and analyse strings that match specified regex patterns. By constructing Python scripts that implemented random choices within the bounds of the given expressions, we produced valid strings that adhered to the constraints of literal characters, grouped selections, and quantifiers.

In summary, this laboratory work highlighted the fundamental importance of regular expressions in software development and data analysis. It showcased the power of computational tools in pattern recognition and the generation of test data, and it reinforced the conceptual knowledge through hands-on coding practice. The exercises completed herein stand as a testament to the integral role that regular expressions play in the broader context of computer science and programming.