

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Inginerie Software și Automatică**

**Tabanschi Nichita FAF-222**

**Report**

*Laboratory work n.4*

***of Limbaje Formale și Automate***

Checked by:

**Dumitru Cretu**, *university assistant*

DISA, FCIM, UTM

**Chișinău – 2024**

### Course: Formal Languages & Finite Automata

# Objectives:

1. Write and cover what regular expressions are, what they are used for;
2. Below you will find 3 complex regular expressions per each variant. Take a variant depending on your number in the list of students and do the following:

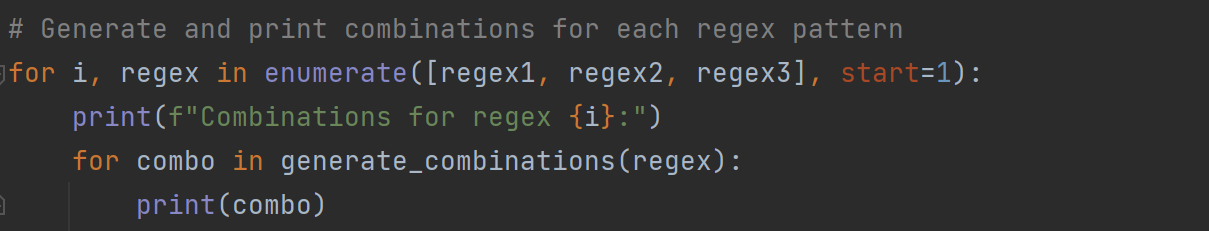
a. Write a code that will generate valid combinations of symbols conform given regular expressions (examples will be shown).

b. In case you have an example, where symbol may be written undefined number of times, take a limit of 5 times (to evade generation of extremely long combinations);

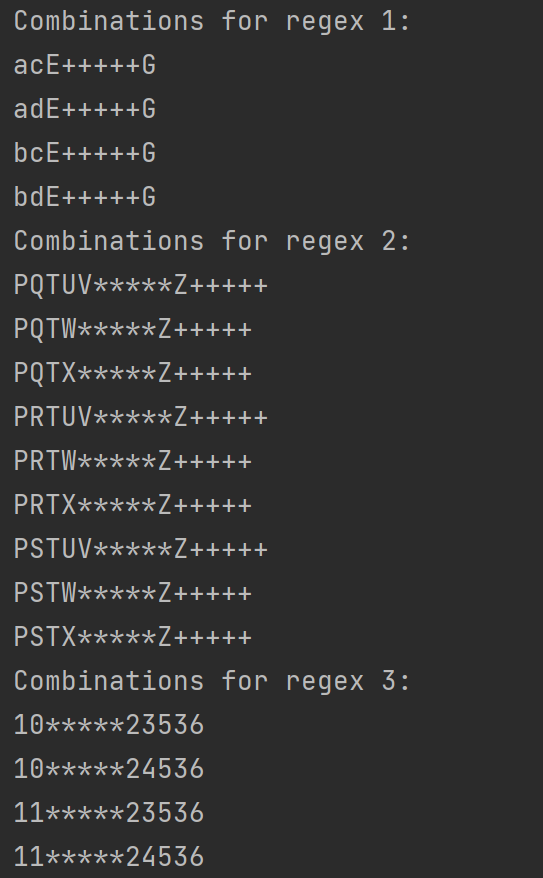
c. **Bonus point**: write a function that will show sequence of processing regular expression (like, what you do first, second and so on)

Write a good report covering all performed actions and faced difficulties.

****

****

**Output:**

****

1. **Regular Expressions**: Three regular expressions are defined at the beginning of the script:
   * **regex1**: "(a|b)(c|d)E+G?"
   * **regex2**: "P(Q|R|S)T(UV|W|X)\*Z+"
   * **regex3**: "1(0|1)\*2(3|4)536"
2. **Function generate\_combinations**: This function takes a regular expression string and an optional limit parameter (default value is 5). It generates combinations for the given regular expression while respecting the limits on repetitions for the '+' and '\*' symbols.
3. **Extracting Variable Parts**: Inside the function, the variable parts of the regular expression are extracted using **re.findall(r'\((.\*?)\)', regex)**. This extracts all substrings enclosed within parentheses.
4. **Generating Combinations**: The **itertools.product** function is then used to generate combinations of these variable parts. Each combination represents a possible substitution for the variable parts in the regular expression.
5. **Substituting Parts**: For each combination, the script iterates over the variable parts and substitutes them into the regular expression using **re.sub**. Additionally, it applies limits on repetitions for the '+' and '\*' symbols by replacing them with their equivalent limited repetitions.
6. **Output**: Finally, the script prints the generated combinations for each regex pattern.

This code is useful for exploring different combinations of regular expressions, especially when dealing with complex patterns and limitations on repetitions. It provides a systematic way to generate and examine various valid combinations based on the given regular expressions.