MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII

AL REPUBLICII MOLDOVA

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

Departamentul Inginerie Software și Automatică

Grebennicova Ecaterina FAF-223

Report

Regular expressions

of Formal Languages & Finite Automata

Checked by:

Cretu Dumitru, *university assistant*

Chișinău – 2023

1. **Theory**

Regular expressions, often abbreviated as regex, are a powerful tool used in programming to search, match, and manipulate text. They allow you to define a search pattern using a sequence of characters. These patterns can specify a set of strings that match a particular syntactic rule, making them incredibly useful for validating text formats (like email addresses or phone numbers), searching within texts, replacing parts of texts, and splitting strings based on specific patterns.

2. **Objectives**

- Write a code that will generate valid combinations of symbols conform given regular expressions
- In case you have an example, where symbol may be written undefined number of times, take a limit of 5 times

3. **Implementation Description**

In this Java program we generate a random string that matches a specific pattern described by a custom regular expression-like format and then print it. The code doesn't use actual regular expression functionality for generation but instead manually constructs strings that would match the described pattern for our cases specifically as it is an easy way to obtain the needed output.

The code:

```
    String combination1 = generateFirstRegex();
    System.out.println("1. (a|b)(c|d)E^+G:");
    System.out.println(combination1);
```

This code snippet represents the main method, where inside we call another function which responds to string generation for the first regex “generateFirstRegex()” and later print it on the screen.

```
  private static String generateFirstRegex() {
List<String> results = new ArrayList<>();

for (String a : new String[]{"a", "b"}) {
    for (String b : new String[]{"c", "d"}) {
        for (int repetitions = 1; repetitions <= 5; repetitions++) {
            results.add(a + b + "E".repeat(repetitions) + "G");
        }
    }
}
```

```
    return selectRandom(results);
}
```

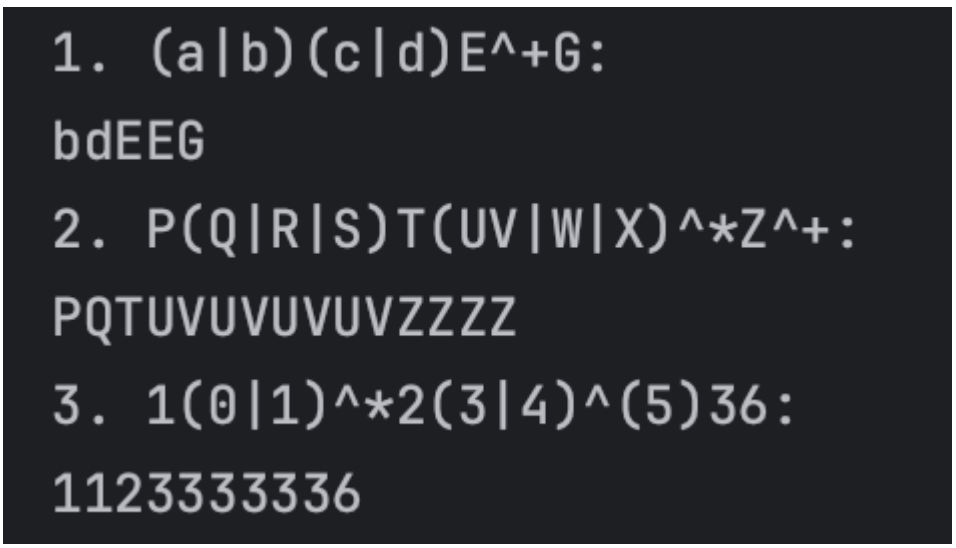
This code snippet represents the method which is called for first regex string generation.

This method generates strings that match the pattern (a|b)(c|d)E^+G, where ^+ showing that the repetitions that can be used for string generation in number for 1 to unlimited. Based on the code's logic, we iterate over two loops, one for a or b and another for c or d, effectively covering all combinations of these two pairs of letters. For each combination of a or b with c or d, we append E repeated from 1 to 5 times, followed by G at the end of the string. All these combinations are added to a list of results which later are called with another method.

```
private static String selectRandom(List<String> list) {
    return list.get(random.nextInt(list.size()));
}
```

This method takes the list of results obtained from the previous method and randomly takes 1 element of it, 1 randomly generated string. For example, possible outcomes of the generateFirstRegex method could include strings like acEEG, bdEEEEG, or bcEG, depending on the random choices made during execution.

Output:



4. Conclusion

In conclusion, I could say that this laboratory work was an easy one where not a lot of difficult code was needed compared with previous laboratories. As well, I could say that I didn't use actual regular expression functionality for generation but instead

manually constructed strings that would match the described pattern with a concrete logic. It was not specified how the laboratory work has to be solved, this is why I consider my approach to be an easy one.

Implementing regularExpression class, I achieved a programmatic solution to generate strings that conform to a predefined pattern. For the first regex, I focused on generating strings that start with either 'a' or 'b', followed by 'c' or 'd', followed by one to five repetitions of 'E', and ending with 'G'. In order to achieve strings we used several loops, lists, randomness to generate specific patterns of the text. The same scenario is repeated for another 2 variants of regular expressions.