Ministerul Educației și Cercetării al Republicii Moldova Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Laboratory work 4:

Regular expressions

Elaborated:

st. gr. FAF-223 Ciornii Alexandr

Verified:

asist. univ. Dumitru Cretu

Chişinău - 2024

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:

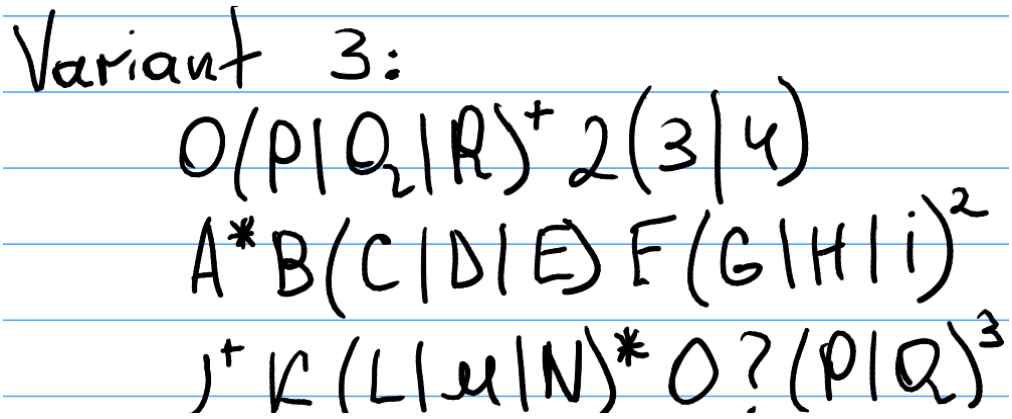
a. Write a code that will generate valid combinations of symbols conform given regular expressions (examples will be shown).

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.

Variant 3:



Examples of what must be generated:

{OPP23, OQQQQ24, ...} {AAABCFGG, AAAAAABDFHH, ...} {JJKLOPPP, JKNQQQ, ...}

Implementation description

I have MainClass and Main method where I simply call all necessary functions. These three functions are in RegEx class,

basically they have very same Idea so I'll explain only third one. Firstly I introduce few local variables that help with string building (note that I use string builder because it's better option for frequently changed string)

```
    StringBuilder sb = new StringBuilder();
    var str = "";
    var rand = new Random().Next(1,6);
```

Then I use for loop that iterates exactly rand number of times, note that rand is limited from 1 to 5, + sign after J means at least one and I limited it to 5 to exclude infinite\too long strings

```
      for (int i = 0; i < rand; i++)
{
    sb.Append('J');
    Console.Write(sb + "  ");
}
```

Next I randomly choose one of 3 symbols and then append it to string 0 to 5 times, star means from zero to infinity, but I limited it for same reason

```
      rand = new Random().Next(3);
switch (rand)
{
    case(0):
        str = "L";
        break;
    case(1):
        str = "M";
        break;
    case(2):
        str = "N";
        break;
    default:
        throw new Exception();
}
rand = new Random().Next(6);
for (int i = 0; i < rand; i++)
{
    sb.Append(str);
    Console.Write(sb + "  ");
}
```

Then I simply coinflip if I should append O symbol (? sign literally means 0 or 1)

```
      rand = new Random().Next(2);
if (rand == 1)
{
    sb.Append('O');
    Console.Write(sb + "  ");
}
```

And lastly I randomly choose P or Q sign and just append it 3 times straightforward

```
      rand = new Random().Next(2);
switch (rand)
{
```

```
    case(0):
        str = "P";
        break;
    case(1):
        str = "Q";
        break;
    default:
        throw new Exception();
}
sb.Append(str);
Console.Write(sb + "  ");
sb.Append(str);
Console.Write(sb + "  ");
sb.Append(str);
Console.Write(sb + "  ");
```

Well, to be fair lastly I return that string, so I could write it in Console note that all `Console.Write(sb + "  ");` lines are for bonus point, hope it works)

Conclusions and Results

Here’re few examples of executed program step-by-step adding symbols till the end

```
O  OR  ORR  ORRR  ORRRR  ORRRRR  ORRRRR2  ORRRRR23  ORRRRR23
A  AB  ABD  ABDF  ABDFG  ABDFGG  ABDFGG
J  JJ  JJJ  JJJJ  JJJJJ  JJJJJK  JJJJJKM  JJJJJKMM  JJJJJKMMM  JJJJJKMMMO  JJJJJKMMMOQ  JJJJJKMMMOQQ  JJJJJKMMMOQQQ
```

```
O  OR  ORR  ORRR  ORRRR  ORRRR2  ORRRR23  ORRRR23
A  AA  AAA  AAAA  AAAAA  AAAAAB  AAAAABD  AAAAABDF  AAAAABDFG  AAAAABDFGG  AAAAABDFGG
J  JJ  JJK  JJKL  JJKLO  JJKLOP  JJKLOPP  JJKLOPPP
```

```
O  OP  OP2  OP23  OP23
A  AA  AAA  AAAB  AAABD  AAABDF  AAABDFI  AAABDFII  AAABDFII
J  JJ  JJJ  JJJK  JJJKL  JJJKLQ  JJJKLQQ  JJJKLQQQ
```

```
O  OR  ORR  ORRR  ORRR2  ORRR23  ORRR23
A  AA  AAA  AAAA  AAAAA  AAAAAB  AAAAABE  AAAAABEF  AAAAABEFH  AAAAABEFHH  AAAAABEFHH
J  JK  JKM  JKMM  JKMMO  JKMMOQ  JKMMOQQ  JKMMOQQQ
```

This lab introduced us to regular expressions (regex) and their practical use in software development. Regex is a powerful tool for pattern matching in strings, enabling concise and flexible text manipulation based on specific patterns.

We developed a program to generate valid combinations of symbols conforming to given regex patterns. Randomization was used to simulate the generation process while ensuring adherence to the patterns. We also limited symbol repetition to five occurrences to prevent excessively long combinations.

A challenge was balancing randomness with pattern adherence during implementation. We included a bonus feature to display the regex processing sequence, aiding understanding of regex operations.

In summary, this lab provided hands-on experience in implementing and understanding regex, essential for software engineers in text processing, data validation, and pattern recognition tasks.