Problem

Check if the string belongs to the given regular expression or not.

Example

If Regualr expressions:

- (0|1)*|0*1*
- (1|0)*(101)+(1|0)*(001)*

then tested Strings:

- 0101010101010101010
- 0011101101101000101010001

Idea used

- For this problem we used java builtin classes **Pattern** and **Matcher**.
- Pattern class is used to compile the regular expression.

```
// Pattern pattern = Pattern.compile(regular expression);
```

Matcher class is used to check if string to be checked matches the pattern.

```
// Matcher matcher = pattern.matcher(test String);
```

Implementation

- Define a test string to be checked.
- Take regular expressions and compile them as a pattern
- Define a matcher of that pattern.
- See if the pattern matches or not using if-else condition. And output accordingly. i.e,

```
Eile Edit View Search Terminal Help

[ceasors@zeronepal src (ceasors)]$ javac RegularExplab1.java
[ceasors@zeronepal src (ceasors)]$ java RegularExplab1
Defined Regular expression: (1|0)*(101)+(1|0)*(001)*
The string 0011101101101000101010001 matches with the expression!
!!

Defined Regular expression: (a*b*|(def)+|a*d+e)+
The string aaabbdefdefaaaaaadde matches with the expression!!!

Defined Regular expression: ((a|b)*(c|d)*)+|ab*c*d
The string abaaaaaaabaccccccccccdacd matches with the expression!
!!
[ceasors@zeronepal src (ceasors)]$
```

Sample Output Screen Shot for problem: Lab1

Problem

Design a scanner that could generate tokens out of a java file i.e, a source code.

Idea used

- The program is designed to analyse the tokens that are contined in a java code so the final output is the list of tokens found in a source file grouped as identifiers, symbols, keywords, digits.
- The program is fed the source code the program is made up of.
- The java's inbuilt class Scanner is also imported in this problem.

Implementation

- Read the inputs source file up to the end of file. Here, a source code of the porgram itself is a source to the program to find out tokens.
- Read a line at a time
- Split each lexeme seperated by the space.
- Check each lexeme if it could be a keyword, a identifier, a digit or a symbol.

```
[ceasors@zeronepal src (ceasors)] $ javac RegularExplab2.java; java RegularExplab2

Keywords found
import import import import import public class private for while do int float char double static switch case import public private class protected return true false if try catch finally else continue break private private private public public for if return true return false public try catch try while for if else int int while if else if continue for int while if int if if else if catch catch for for for int for char public static

identifiers found
java aux SystemColor java io File java io FileNotFoundException java util ArrayList java util NoSuchElementException java util Scanner RegularExplab
String keywords Scanner input ArrayList String keywords found ArrayList Integer identifiers found new ArrayList String symbols found new ArrayList Integer identifiers found new ArrayList String symbols found new ArrayList Doctore Boolean isKeyword String token String keywords token equals keyword to state gonung in the user entered as the argument input new Scanner new File RegularExplab java FileNotFoundException fileNotFoundException System err println Error opening file System exit String sentence input healthies String in Stems out println String sentence input healthies String since out token charat digit token charat digit token length is token length is token length java lang Character isAlexiter token charat is System out println is start i digits found add digit token length is the information of the println System out println S
```

Sample Output Screen Shot for problem: Lab2

Problem

Read input strings from a file and check to see if they belong to the given regular expression or not.

Idea used

- Store input strings in a file in different line
- Check for matches for each string using 'String.matches(Regex r)' function.

Implementation

- Read the inputs strings from the file one line at a time.
- Run a loop for all these lines.
- Inside the loop check for matches.
- If they match notify the user that they belong.

```
[ceasors@zeronepal src (ceasors)]$ cat expression.txt
1*
ab*c
[ceasors@zeronepal src (ceasors)]$ java RegularExplab3
expression 1*
The string does not matches with the expression!!!
expression ab*c
The string ac matches with the expression!!!
[ceasors@zeronepal src (ceasors)]$
```

Sample Output Screen Shot for problem: Lab3

Problem

Remove the immediate left recursion of a grammer with general form.

Idea

- Determine the alpha, beta from the input string. [General form A->A[(alpha) | (beta)]
- Then change the form into: A->(beta)A'
 A'->(alpha)A' | (Epsilon)

Implementaion

- Get the input from user.
- Check to see if the input is left recursive or not.
 - o This is done by comparing the initial value in the array and the one after the '>'symbol.

If recursive then continue with the following steps:

- Since the program is designed for general form so first of all we chunked the input production into a jagged array holding seperate possible production rules seperated by the or '|' symbol. For this we counted the number of '|' symbols present in the production and made a for loop run for number of '|' symbols present plus 1. By this jagged array we could easily handel the all input productions and check if immediate left recursion exists.
- Then using a for loop we iterate over those chunks to adjust alpha or beta case of input. For this we implemented two array dynamically created using malloc and our sample output looks like this:

```
Enter the grammer:<length limit 20>
The grammer must be formated as: S=>Sabc|x0z|Sega|hello|world
E->Egg|ball|East|Echo|apple|hello|Eworld
left recusion exists...
***So after chunking the input, we get
        Е
                g
                        g
        b
                a
        Е
                а
                        S
        Е
                                 0
        a
                        p
                                         е
                е
                                         0
The required expression is:
E=> ballE'|appleE'|helloE'
E'=> qqE'|astE'|choE'|worldE'|E[ceasors@zeronepal src (ceasor:
```