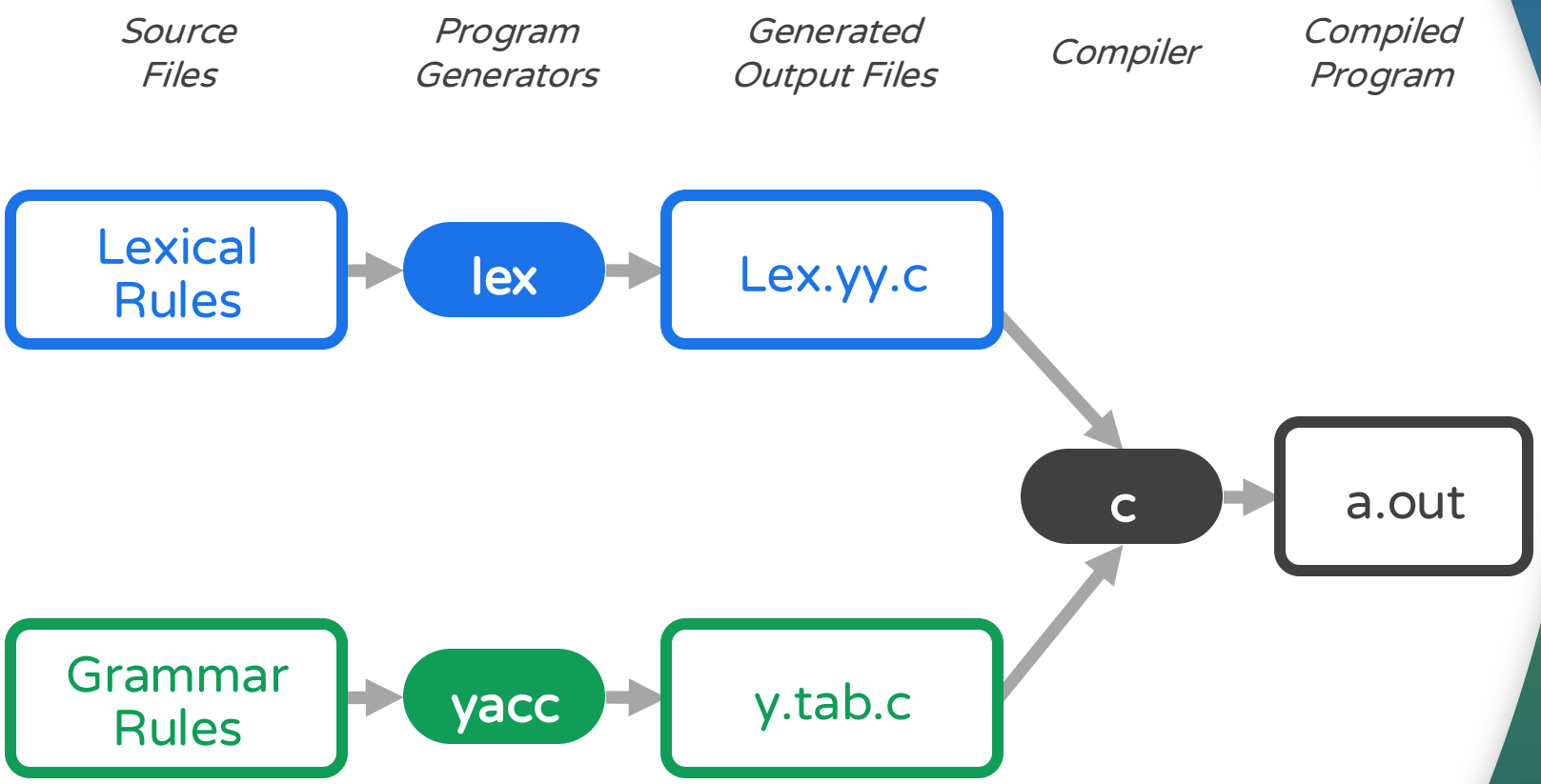
國立中山大學 National Sun Yat-sen University
資訊工程學系 Department of Computer Science and Engineering

Compiler

Lex Tutorial

TA| Xiang, Long, Ding

Division of Work for the Compiler



Lex work

Operation

Lex treats input as a sequence of characters.

A sequence of consecutive characters forms a token.

Purpose

Lex's purpose is to check the validity of tokens, such as invalid variable names (identifiers).

Condition

Lex requires predefined rules:
Regular Expressions which is can be used to identify tokens.

Lex Input

Taking Java as an example

```
public static void main() {
    int c;
    int a = 5;
    int 5a;       //Invalid identifier

    c = add(a, 10);
    if (c > 10)
        print("c = " + -c);
    else
        print(c);
    print("Hello World");
}
```

Lex Format

Divided into three parts,
each separated by %%.

```
Definition
%%
Lex Rules
%%
User code
```

Definition

demo.l

```
%{
#include <stdio.h>
unsigned charCount=1, idCount=0, lineCount=1;
%}
operator [\+\-\*\/]
space [ \t]
eol \n

/* You should write your own regular expression. */
reserved_word
symbol
id

%%
```

Rules


```
%%

{operator} {
    printf("Line: %d, 1st char: %d, \"%s\" is an
            \"operator\".\n", lineCount, charCount, yytext);
    charCount += yyleng;
}
{space} {
    charCount++;
}
{eol} {
    lineCount++;
    charCount = 1;
}
{reserved_word} {
    /* You should write your own code */
}

%%
```

Rules

- The priority order of the rules matched by the Scanner:
  - It will scan the longest token for matching.
  - If the matching lengths are the same, it will consider the order in which they were defined (from top to bottom).

User Code

demo.l

```
%%
int main(){
    yylex();
    return 0;
}
```

Let's take a look at the Test Cases

Test File

test1.java

```java
public class Test1 {
    public static int add(int a, int b) {
        return a + b;
    }
}
```

Output

```
bot@Pc-Nsysu-Lab-Curtis ~/compiler/lab1/LexDemo
$ ./demo < test1.java
Line: 1, 1st char: 1,  "public" is a "ReservedWord".
Line: 1, 1st char: 8,  "class" is a "ReservedWord".
Line: 1, 1st char: 14, "Test1" is an "ID".
Line: 1, 1st char: 20, "{" is a "symbol".
Line: 2, 1st char: 5,  "public" is a "ReservedWord".
Line: 2, 1st char: 12, "static" is a "ReservedWord".
Line: 2, 1st char: 19, "int" is a "ReservedWord".
Line: 2, 1st char: 23, "add" is an "ID".
Line: 2, 1st char: 26, "(" is a "symbol".
Line: 2, 1st char: 27, "int" is a "ReservedWord".
Line: 2, 1st char: 31, "a" is an "ID".
Line: 2, 1st char: 32, "," is a "symbol".
Line: 2, 1st char: 34, "int" is a "ReservedWord".
Line: 2, 1st char: 38, "b" is an "ID".
Line: 2, 1st char: 39, ")" is a "symbol".
Line: 2, 1st char: 41, "{" is a "symbol".
Line: 3, 1st char: 9,  "return" is a "ReservedWord".
Line: 3, 1st char: 16, "a" is an "ID".
Line: 3, 1st char: 18, "+" is an "operator".
Line: 3, 1st char: 20, "b" is an "ID".
Line: 3, 1st char: 21, ";" is a "symbol".
Line: 4, 1st char: 5,  "}" is a "symbol".
Line: 5, 1st char: 1,  "}" is a "symbol".
The symbol table contains:
Test1
add
a
b
```

Test File & Output

test1.java

Test File

```
public class Test1 {
    public static int add(int a, int b) {
        return a + b;
    }
}
```

Output

```
bot@Pc-Nsysu-Lab-Curtis ~/compiler/lab1/LexDemo
$ ./demo < test1.java

Line: 1, 1st char: 1,  "public" is a "ReservedWord".
Line: 1, 1st char: 8,  "class" is a "ReservedWord".
Line: 1, 1st char: 14, "Test1" is an "ID".
Line: 1, 1st char: 20, "{" is a "symbol".
```

Lex special characters

- These characters have special meanings in regular expressions. If you want to treat them as ordinary characters, please add a backslash (\) before them (escape character).
  - ? * + | ( ) ^ $. [ ] { } " \

- Digit [0-9]
- Letter [a-zA-Z]
- Operator [\+\-*]

Using Lex

How to use Lex File

- Our goal is to compile `demo.l` into an executable scanner.
- First, we need to install the `flex` program to compile our lex file.
  - sudo apt-get install flex (using Ubuntu as an example)
- The demo.l file is compiled into a C source file by using Flex.
  In which C source file is our scanner.
  - flex demo.l
- The default C source file name is lex.yy.c.
  Finally, we can use gcc to compile it into an executable file:
  - gcc lex.yy.c –o demo -lfl
- The executable file is named demo. Let's assume the file which we want to scan is test1.java.
  - ./demo < test1.java

Regular Expression

Commonly used character symbols

`.`	Any character (excluding line breaks)
`\d`	Any digit
`\D`	Any non-digit
`\w`	Any text, numbers, or underscores
`\W`	Any non-letter, non-number, non-baseline
`\s`	Any blank character (blank, positioned, line break)
`\S`	Any non-whitespace character (whitespace, positioning, line break)
`\n`	Newline character
`\t`	Positioning character (Tab)
`\r`	Carriage return character
`\0`	Null character

Special character symbols

\.	. character
\?	? character
*	* character
\+	+ character
|	| character
\^	^ character
\$	$ character
\”	“character

\(	(character
\)	) character
\[	[character
\]	] character
\{	{character
\}	} character
\\	\ character
\/	/ character

Common enum rules

Rule	Explain	Example
`[12abc]`	A single character of 1, 2, a, b, c	`1, 2, a, b, c`
`[^12abc]`	A single character except 1, 2, a, b, c	`3, 4, d, e, f`
`[0-9A-Z]`	A character in the range of 0-9 or A-Z	`0, 1, A, B, C`
`[ab]|[0-9]`	A single character of a, b, or in the range of 0-9	`a, b, 0, 1, 2`

Commonly used frequency symbols
(added at the end)

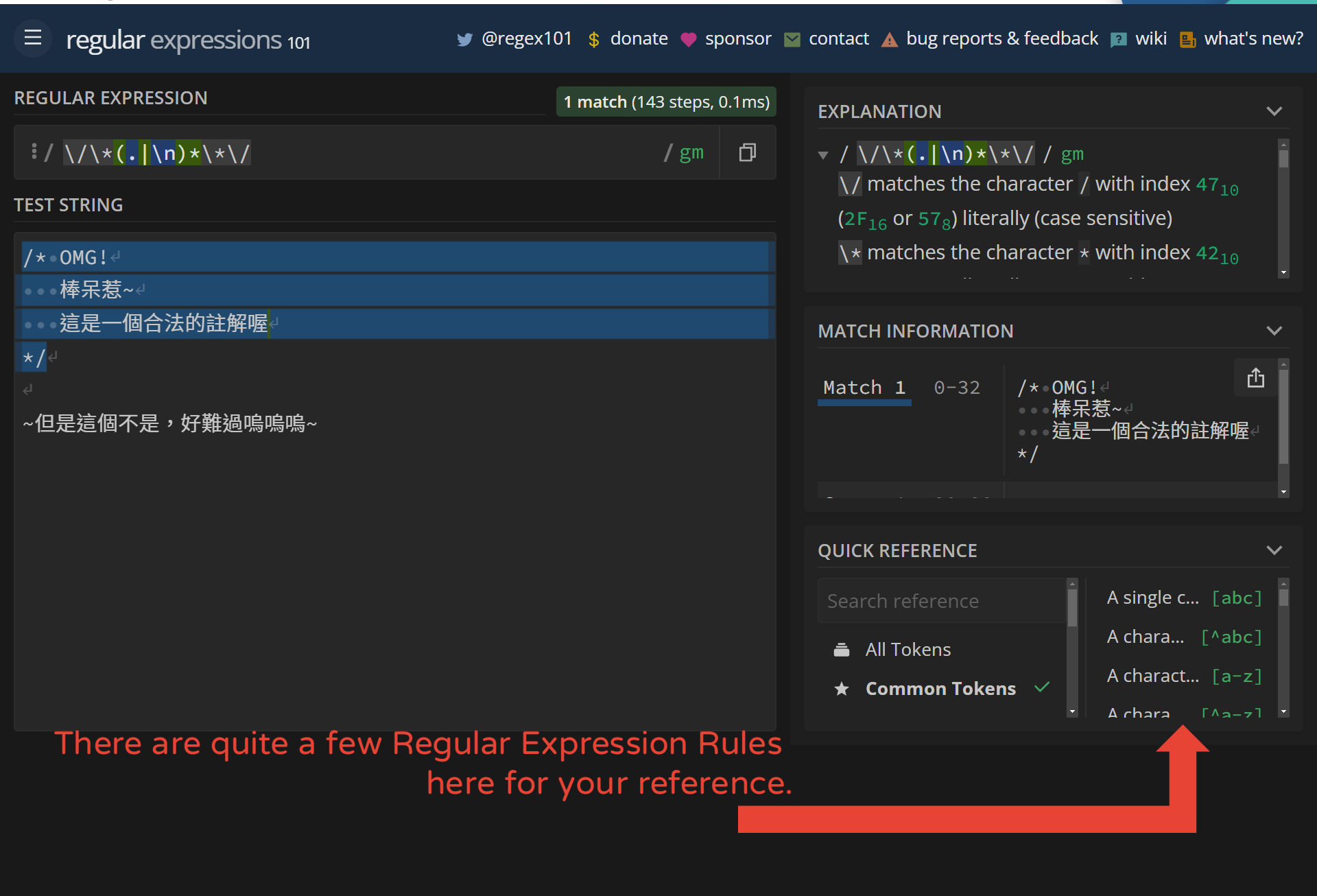
symbol	Explain
*	Repeat 0 ~ ∞ times
+	Repeat 1 ~ ∞ times
?	Repeat 0 ~ 1 times
{n}	Repeat n times
{n,}	Repeat n ~ ∞ times
{n,m}	Repeat n ~ m times

Regular Expression

- If you are not familiar with standard languages, there are abundant resources available online, such as...
  https://www.vixual.net/blog/archives/211

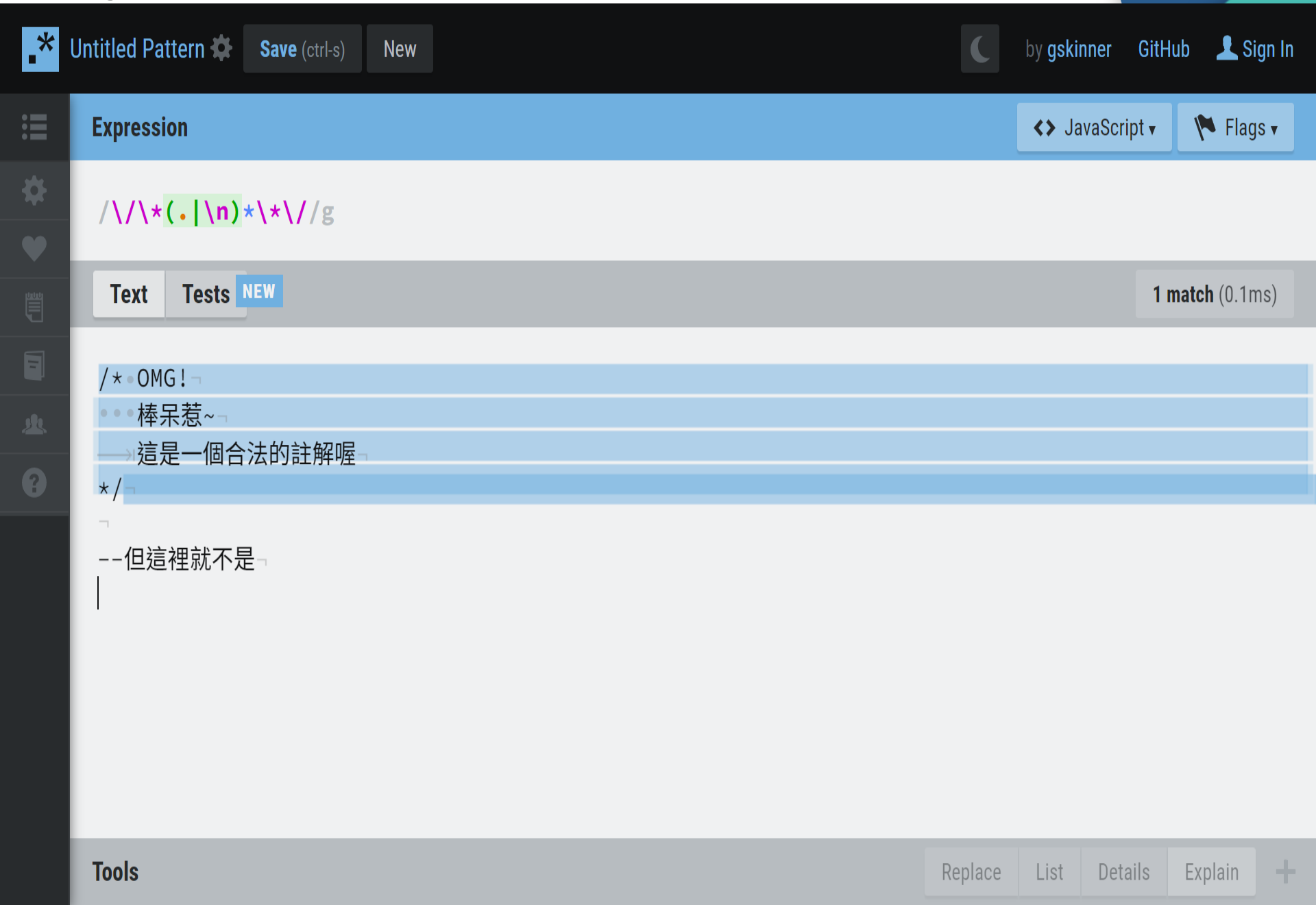
- Online Regular Expression Tester
  - https://regex101.com
  - https://regexr.com

regexr.com



There are quite a few Regular Expression Rules here for your reference.

regexr.com


Untitled Pattern
Save (ctrl-s)
New
by gskinner
GitHub
Sign In
Expression
JavaScript
Flags
/\/*(.|\n)**\//g
Text
Tests
NEW
1 match (0.1ms)
/* OMG!
棒呆惹~
這是一個合法的註解喔
*/
--但這裡就不是
Tools
Replace
List
Details
Explain

About Homework 1

Recommended env

- Install Ubuntu on a virtual machine
  - Ubuntu 22

Homework Submission Instructions

- **DUE DATE: [Date] 23:59**
- **The program demo environment is Ubuntu 22.04.2 LTS, so please ensure your code can compile and execute.**
- **Please refer to the test files on the course webpage to verify your program.**
- **The teaching assistant will design additional test files, so please ensure your Regular Expressions match most cases.**
- **For example, complex variable names, floating-point numbers must be negative, etc.**
- **Please submit your assignment on time. A 30% discount will be applied for each day late.**
- **Please compress your assignment into a single file and upload it to the online university. Name the file "Student ID_hw1.zip". Incorrect student ID will result in a -10 grade; no student ID will result in a -50 grade.**
- **A demo session will be scheduled after the submission deadline. Please arrive at the EC5023 Database Systems Lab on time to find the teaching assistant for the demo.**

Contact Information

Feel free to ask the teaching assistant questions.

丁襄龍
clovedragon12@gmail.com
EC5023 DBSL


Q&A