LAB REPORT

CSE232: Compiler Design Lab

02
[Report Number]

Topic: Write a Flex program to recognize the following types of strings.

Submitted To

Mushfiqur Rahman Chowdhury (MRC)

Lecturer

Department of CSE, Daffodil International University

Submitted By

Student ID: 0242220005101781

Section: 63 N

Student Name: SHAHARIAR HOSSAIN

Date of Assignment Distribution: 10 November 2025

Date of Assignment Submission: 22 November 2025

Experiment No: 02	Mapping: CO1 and CO2

Experiment Name	Write a Flex program to recognize the following types of strings.

Experiment Details:

Write a Flex program to recognize the following types of strings:
❖ Strings that consist of one or more occurrences of 'x'. [x+]
❖ Strings that consist of one or more 'x' followed by one or more 'y'. [x+y+]
❖ Strings that match exactly 'x', 'y', 'z' in that order. [xyz]

Lexer.l:

```
%{
#include <stdio.h>
#include <stdlib.h>

/* Declarations used when scanning a string with Flex */
extern FILE *yyin;
/* Use the scanner's buffer type (defined in generated lex.yy.c). */
struct yy_buffer_state;
extern struct yy_buffer_state *yy_scan_string(const char *);
extern void yy_delete_buffer(struct yy_buffer_state *);
%}

%option noyywrap

%%

x+                  { printf("Matched: %s (One or more 'x')\n", yytext); }
x+y+                { printf("Matched: %s (One or more 'x' followed by one or more
'y')\n", yytext); }
xyz                 { printf("Matched: %s (Exactly 'x', 'y', 'z' in order)\n",
yytext); }

.|\n                { /* Ignore any other characters or newlines */ }

%%

int main(int argc, char *argv[]) {
    if (argc > 1) {
        /* Try opening argv[1] as a file first. If that fails,
           treat argv[1] as a literal input string. */
        FILE *f = fopen(argv[1], "r");
        if (f) {
            yyin = f;
            yylex();
            fclose(f);
        } else {
            YY_BUFFER_STATE bp = yy_scan_string(argv[1]);
            yylex();
            yy_delete_buffer(bp);
        }
    } else {
        /* No argument: read from stdin */
        yylex();
    }
    return 0;
}
```

<u>tests.txt:</u>

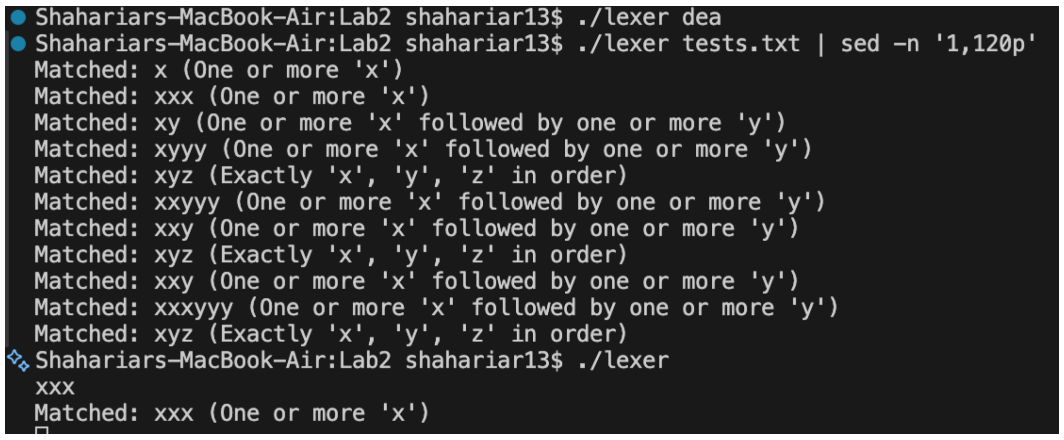
```
x
xxx
xy
xyyy
xyz
xxyyy
xxyz
xyzzz
abc
def
xxy
xxxyyy
xyzabc
yyy
zzzz
```

<u>Command:</u>

```
flex lexer.l
clang lex.yy.c -o lexer
./lexer tests.txt        # File argument
./lexer < tests.txt      # stdin redirect
./lexer
```

Obtained Output:

	Desired Output?

	YES

```
Shahariars-MacBook-Air:Lab2 shahariar13$ ./lexer dea
Shahariars-MacBook-Air:Lab2 shahariar13$ ./lexer tests.txt | sed -n '1,120p'
Matched: x (One or more 'x')
Matched: xxx (One or more 'x')
Matched: xy (One or more 'x' followed by one or more 'y')
Matched: xyyy (One or more 'x' followed by one or more 'y')
Matched: xyz (Exactly 'x', 'y', 'z' in order)
Matched: xxyyy (One or more 'x' followed by one or more 'y')
Matched: xxy (One or more 'x' followed by one or more 'y')
Matched: xyz (Exactly 'x', 'y', 'z' in order)
Matched: xxy (One or more 'x' followed by one or more 'y')
Matched: xxxyyy (One or more 'x' followed by one or more 'y')
Matched: xyz (Exactly 'x', 'y', 'z' in order)
Shahariars-MacBook-Air:Lab2 shahariar13$ ./lexer
xxx
Matched: xxx (One or more 'x')
```

Alternative Steps/Solution (If any):

I have implemented a Flex lexer that can take an input file or command-line string and recognize the patterns 'x+', 'x+y+', and 'xyz'.

Observation/ Comments:

The Flex lexer successfully identifies the patterns `'x+'`, `'x+y+'`, and `'xyz'` from both input files and command-line strings, demonstrating its ability to process different input formats and recognize the specified patterns.

Appendix A: Course Outcomes, Complex Engineering Problems (EP) and Complex Engineering Activities (EA) Addressing.

Table: CSE312 Course Outcomes (COs) with Mappings

COs	CO Statements	POs	Learning Domains	Knowledge Profile	Complex Engineering Problem	Complex Engineering Activities
CO 1	Demonstrate a comprehensive understanding of fundamental database management concepts, including the relational data model, normalization techniques, and SQL basics.	PO1	C2 A2 P2	K2 K3 K4 K8	EP1 EP4	
CO 2	Design, implement and optimize relational databases, incorporating advanced SQL queries, indexing techniques and query optimization strategies.	PO3	C3 A3 P3	K2 K3 K4 K6 K8	EP1 EP2 EP7	EA3

CO 3	Understand and Analyze security measures, distributed database architectures and emerging trends in database management, demonstrating an understanding of the broader context and challenges in the field.	PO5	C4 A4 P3	K6	EP4	

Table: Addressing CO (1 to 3), Knowledge Profile (K), Attainment of Complex Engineering Problems (EP):

SN	Engineering Problem (EP) Definition	Attain Ment	CO	Justification (with Knowledge Profile)
01	EP1: Depth of Knowledge required	Yes/No	CO1, CO2	
02	EP2: Range of Conflicting Requirement s	Yes/No	CO2	
03	EP4: Familiarity of Issues	Yes/No	CO1, CO3	
04	EP7: Interdependence	Yes/No	CO2	

Table: Addressing COs

SN	COs	Attainment	Justification
01	CO1	Yes/No	These Lab activities attain CO1 by.
02	CO2	Yes/No	N/A
03	CO3	Yes/No	These Lab activities attain CO3 by.

Table: Lab-Wise Recommended Topics

Lab Class No.	**Proposed Activity**
Lab 1	Write a Flex program to recognize the following types of strings.
Lab 2	Write a Flex program to recognize the following types of strings.
Lab 3	Write a calculator using Flex and Bison that supports