Compiler Design LAB (CIC-351)

Practical File

Faculty Name : Ms. Sakshi Jha

Student Name : Amit Singhal

Roll No : 11614802722 (C-6)

Semester : 5th Semester

Group : 5C6 (CSE Shift-1)

Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi - 110086

Maharaja Agrasen Institute of Technology

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION

"To attain global excellence through education, innovation, research, and work ethics in the field of Computer Science and engineering with the commitment to serve humanity."

MISSION

M1 To lead in the advancement of computer science and engineering through internationally recognized research and education.

M2 To prepare students for full and ethical participation in a diverse society and encourage lifelong learning.

M3 To foster development of problem solving and communication skills as an integral component of the profession.

M4 To impart knowledge, skills and cultivate an environment supporting incubation, product development, technology transfer, capacity building and entrepreneurship in the field of computer science and engineering.

M5 To encourage faculty, student's networking with alumni, industry, institutions, and other stakeholders for collective engagement.

Department of Computer Science & Engineering

VISION

"To attain global excellence through education, innovation, research, and work ethics in the field of Computer Science and engineering with the commitment to serve humanity."

MISSION

M1 To lead in the advancement of computer science and engineering through internationally recognized research and education.

M2 To prepare students for full and ethical participation in a diverse society and encourage lifelong learning.

M3 To foster development of problem solving and communication skills as an integral component of the profession.

M4 To impart knowledge, skills and cultivate an environment supporting incubation, product development, technology transfer, capacity building and entrepreneurship in the field of computer science and engineering.

M5 To encourage faculty, student's networking with alumni, industry, institutions, and other stakeholders for collective engagement

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

MAIT

Lab Assessment Sheet

S.No	Experiment	Date	Marks					Total Marks	Signature
			R1	R2	R3	R4	R5		

Lab Assessment Sheet

S.No	Experiment	Date	Marks					Total Marks	Signature
			R1	R2	R3	R4	R5		

PROGRAM – 1

❖ AIM :: Basic program to practice flex, bison & lex.yy.c

Theory ::

Flex:

- **Purpose:** Tool for generating lexical analyzers.
- **Input:** .l files (define lexical rules and patterns).
- **Output:** C source file (lex.yy.c), which can be compiled into an executable.

Bison:

- **Purpose:** Parser generator for processing grammar and parsing tokenized input.
- **Works with Flex:** Bison creates a parser that processes tokens generated by the Flex lexer.

Dev C++:

- **Purpose:** IDE for compiling and debugging C programs.
- **Usage:** Compiles the lex.yy.c file produced by Flex into an executable (a.out).
- **Features:** Provides an integrated environment for editing, compiling, and debugging generated C code.

Key Concepts:

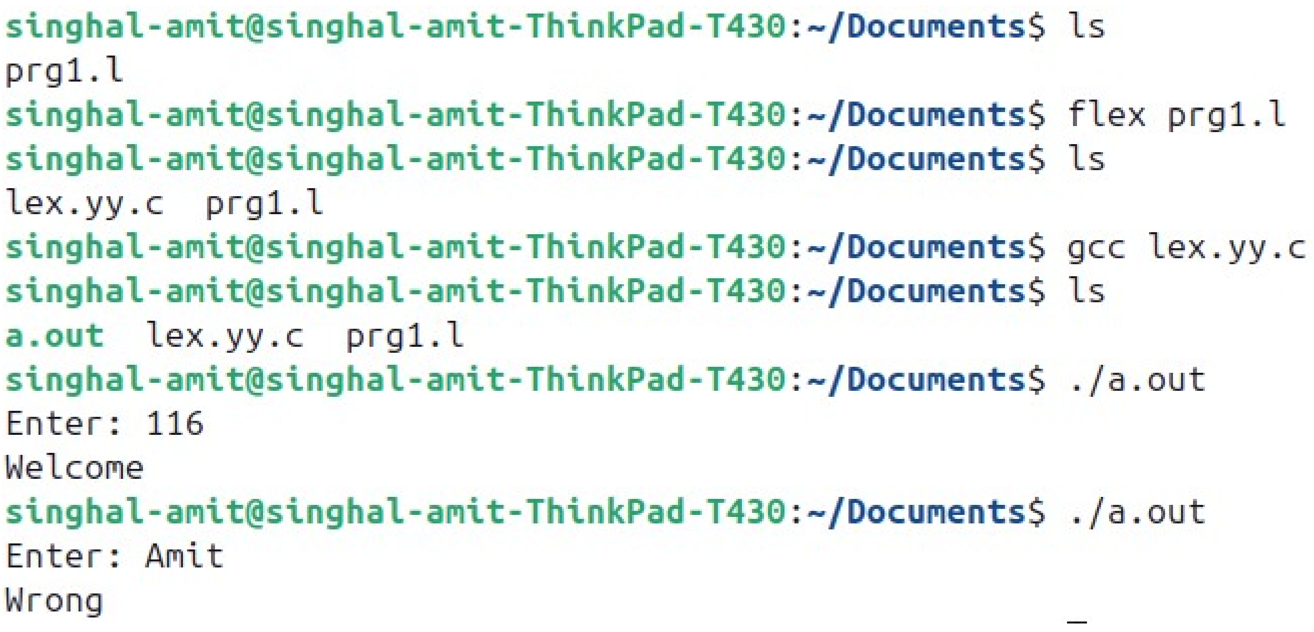
1. **.l File:** The input file for Flex, containing lexical rules and patterns.
2. **lex.yy.c:** C file generated by Flex, containing the lexical analyzer.
3. **a.out:** Default name of the compiled executable generated after compiling lex.yy.c.
4. **Basic Syntax in Lex:**
   - **Definitions Section:** %{ ... %} for including C code.
   - **Rules Section:** Pattern matching rules (regex) and actions.
   - **Main Section:** yylex() to process input, yywrap() to indicate end of input.

Lex Code ::

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

%%
```

```
[0-9]     { printf("Welcome\n"); exit(0); }
.         { printf("Wrong\n"); exit(0); }

%%

int yywrap() { return 1; }

int main() {
    printf("Enter: ");
    yylex();  // Start the lexer
    return 0;
}
```

Output ::



```
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ ls
prg1.l
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ flex prg1.l
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ ls
lex.yy.c  prg1.l
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ gcc lex.yy.c
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ ls
a.out  lex.yy.c  prg1.l
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ ./a.out
Enter: 116
Welcome
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ ./a.out
Enter: Amit
Wrong
```

PROGRAM – 2

❖ AIM :: WAP to check whether a string includes a `keyword` or not.

Theory ::

Keywords in C

In C, keywords are reserved words that have predefined meanings and specific functionalities in the language. They cannot be used for naming variables, functions, or other identifiers, as they are integral to the syntax and structure of C programs. Keywords represent fundamental data types, control structures, memory management, and other language features that are essential for writing C programs.

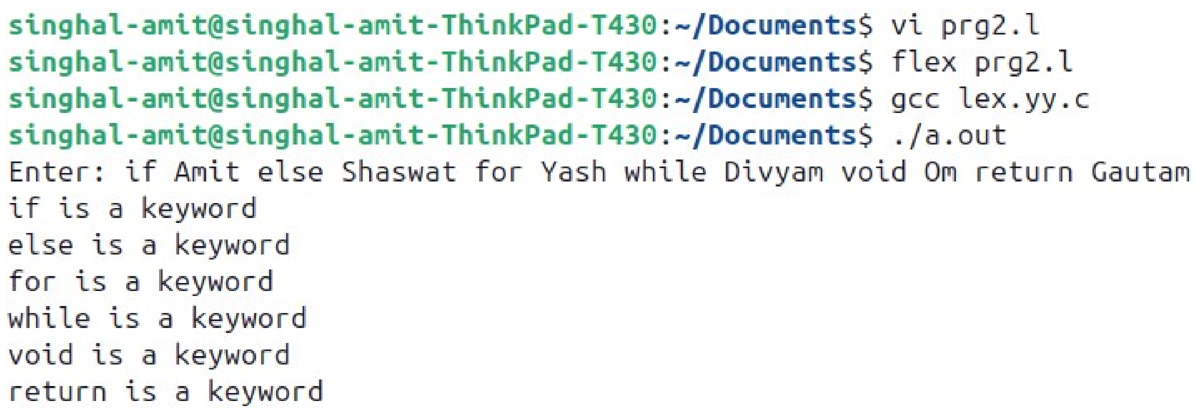
Here is the list of 32 reserved keywords in C:

auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, long, register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, volatile, while

Lex Code ::

```
%{
#include <stdio.h>
#include <string.h>
%}

%option noyywrap

%%
"auto"|"break"|"case"|"char"|"const"|"continue"|"default"|"do"|"double"|"else"|"enum"|"extern" {
   printf("%s is a keyword\n", yytext);
}
"float"|"for"|"goto"|"if"|"int"|"long"|"register"|"return"|"short"|"signed"|"sizeof"|"static"|"struct" {
   printf("%s is a keyword\n", yytext);
}
"switch"|"typedef"|"union"|"unsigned"|"void"|"volatile"|"while" {
    printf("%s is a keyword\n", yytext);
}
.|\n { /* Ignore other characters */ }

%%

int main() {
    printf("Enter: ");
    yylex();
    return 0;
}
```

Output ::



```
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ vi prg2.l
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ flex prg2.l
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ gcc lex.yy.c
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ ./a.out
Enter: if Amit else Shaswat for Yash while Divyam void Om return Gautam
if is a keyword
else is a keyword
for is a keyword
while is a keyword
void is a keyword
return is a keyword
```

PROGRAM – 3

❖ AIM :: WAP to count no. of `tokens` in a string.

Theory ::

Tokens in C

In C programming, a **token** is the smallest unit of a program that has a meaning in the language. Tokens are categorized into several types, including:

1. **Keywords**: Reserved words with predefined meanings (e.g., int, return, if).
2. **Identifiers**: Names given to variables, functions, arrays, etc. (e.g., main, count).
3. **Constants**: Literal values like numbers and characters (e.g., 42, 'a').
4. **Operators**: Symbols representing operations (e.g., +, -, *, /).
5. **Punctuation**: Symbols that help structure the code (e.g., ;, ,, {, }).

Examples of Tokens:

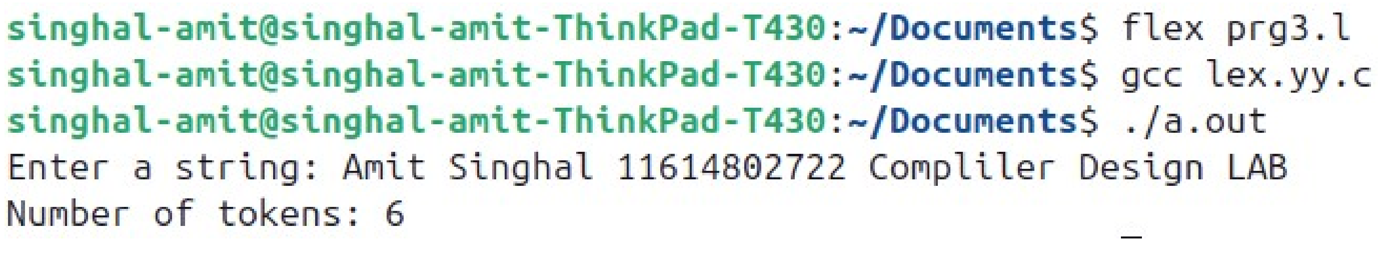
- **Keywords**: if, while, for
- **Identifiers**: variable, sum, total
- **Constants**: 10, 3.14, 'x'
- **Operators**: +, -, =
- **Punctuation**: ;, {, }

Lex Code ::

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

int token_count = 0;
%}

%%
[a-zA-Z_][a-zA-Z0-9_]*     { token_count++; }  // Identifiers
[0-9]+                     { token_count++; }  // Numbers
"+"|"-"|"*"|"/"            { token_count++; }  // Operators
";"|"{"|"}"|"("|")"        { token_count++; }  // Punctuation
```

```
[ \t\n]+                    { /* Ignore whitespace */ }
.                           { /* Ignore other characters */ }
%%

int yywrap() {
   return 1;
}

int main() {
   printf("Enter a string: ");
   yylex();                                          // Start lexing
   printf("\nNumber of tokens: %d\n", token_count);
   return 0;
}
```

Output ::



```
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ flex prg3.l
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ gcc lex.yy.c
singhal-amit@singhal-amit-ThinkPad-T430:~/Documents$ ./a.out
Enter a string: Amit Singhal 11614802722 Compliler Design LAB
Number of tokens: 6
```