Assignment #1 Report
Compilers (Bonus Part)

No	Name	ID
1	زياد احمد ابراهيم زيدان	19015709
2	السيد جاد السيد عبدالرحمن	19015894
3	محمد عبدالعزيز محمد عبدالعزيز	19015941
4	عمر خيرت محمد ابو ضيف	19016063

Introduction:

The lexical analyzer is a crucial component in compiler construction, responsible for breaking down the input source code into tokens. Lex (or its modern counterpart, Flex) is a powerful tool for generating lexical analyzers based on specified regular expressions. This report details the steps required to generate a lexical analyzer for a simple programming language using Lex.

Step 1: Define Regular Expressions:

In the given code snippet, regular expressions are used to describe the patterns of various tokens in the programming language. Each regular expression corresponds to a category of tokens, such as identifiers, numbers, operators, and keywords. These are essential in recognizing and categorizing different components of the source code.

<u>Step 2: Lex Specification:</u>

The Lex tool requires a specification file that contains the regular expressions along with the corresponding actions to be taken when a match is found. In the provided code, the Lex specification is between the %{ and %} markers. Here, actions involve printing the recognized token along with additional information such as the lexeme (e.g., ID: %s, NUM: %s).

<u>Step 3: Token Recognition and Output:</u>

For each regular expression, Lex generates a C program that recognizes tokens based on these patterns. The actions associated with each pattern determine what happens when a match is found. In this case, the program prints the type of token along with its value or lexeme.

<u>Step 4: Using Lex/Flex to Generate the Lexer:</u>

To generate the lexical analyzer, you can use the following command in the terminal:

```
sudo apt-get update

sudo apt-get install flex

lex lex.l

gcc lex.yy.c -o lexer -ll

./lexer < test_program
```

lex lex.l:

This command uses the Lex tool to process the Lex specification file lex.l and generate a C program.

The Lex specification file (lex.l) contains regular expressions and corresponding actions that define the patterns and behavior of the lexical analyzer.

gcc lex.yy.c -o lexer -ll:

gcc: Stands for the GNU Compiler Collection, a compiler system used for compiling C programs.

lex.yy.c: The C source code file generated by Lex from the Lex specification file.

-o lexer: Specifies the output file name of the compiled program as lexer.

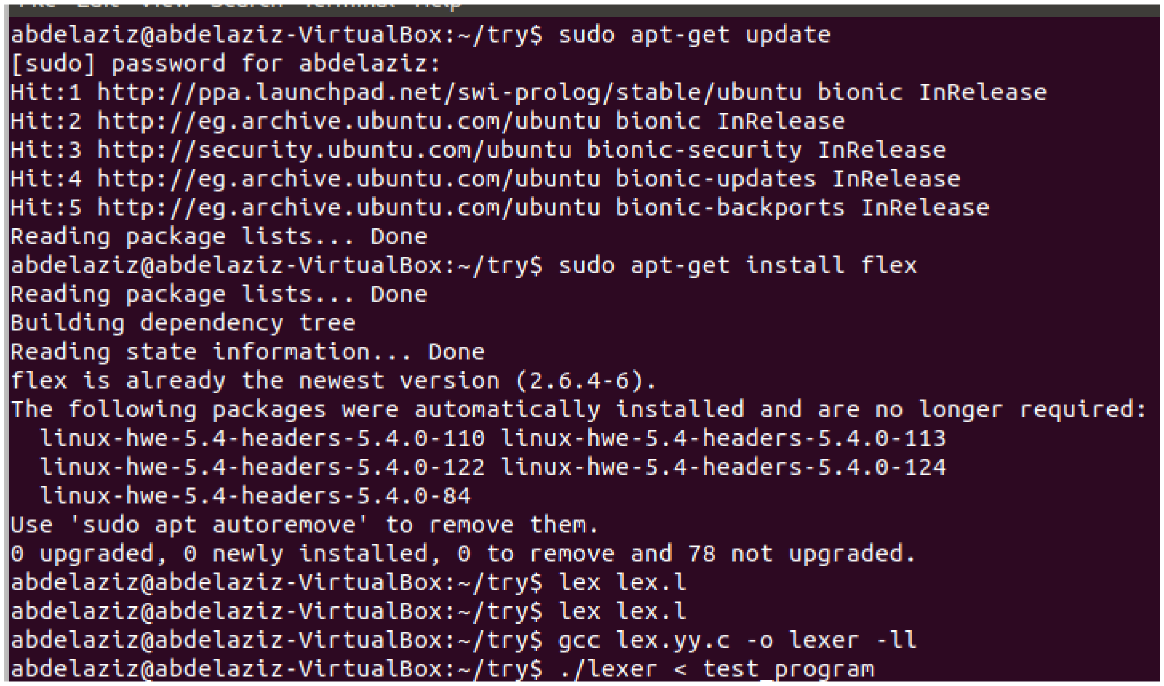
-ll: Links the lexer with the Flex library (libfl.a), which is required for handling the generated lexical analyzer.

./lexer < test_program:

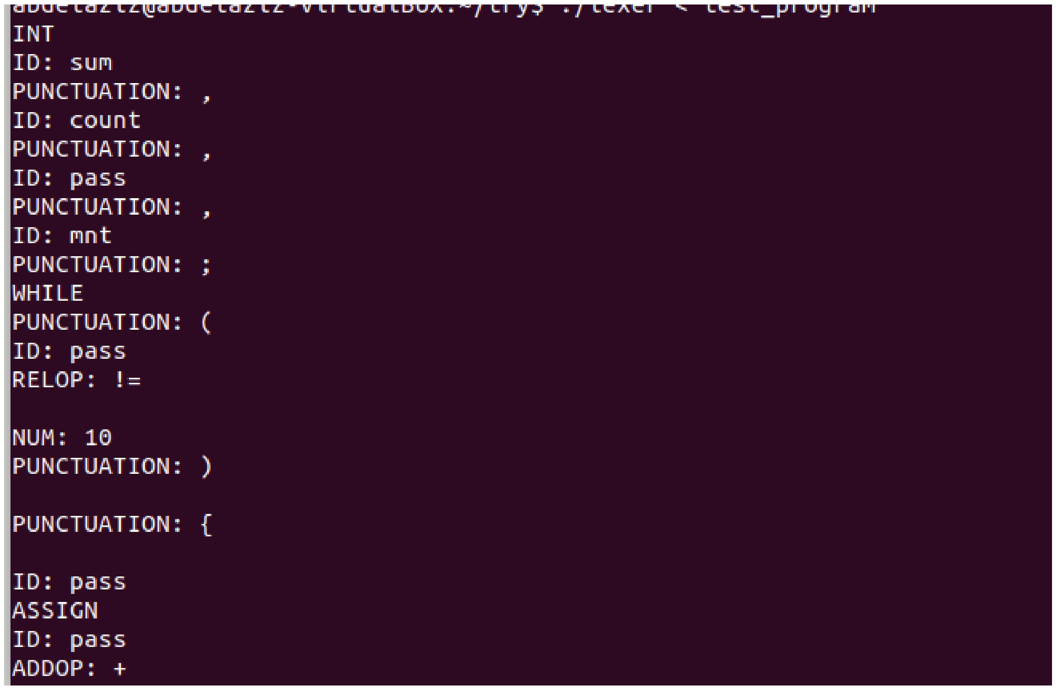
./lexer: Executes the compiled lexer program.

< test_program: Redirects the contents of the file test_program as input to the lexer program

Screenshots:



```
abdelaziz@abdelaziz-VirtualBox:~/try$ sudo apt-get update
[sudo] password for abdelaziz:
Hit:1 http://ppa.launchpad.net/swi-prolog/stable/ubuntu bionic InRelease
Hit:2 http://eg.archive.ubuntu.com/ubuntu bionic InRelease
Hit:3 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:4 http://eg.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:5 http://eg.archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
abdelaziz@abdelaziz-VirtualBox:~/try$ sudo apt-get install flex
Reading package lists... Done
Building dependency tree
Reading state information... Done
flex is already the newest version (2.6.4-6).
The following packages were automatically installed and are no longer required:
  linux-hwe-5.4-headers-5.4.0-110 linux-hwe-5.4-headers-5.4.0-113
  linux-hwe-5.4-headers-5.4.0-122 linux-hwe-5.4-headers-5.4.0-124
  linux-hwe-5.4-headers-5.4.0-84
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 78 not upgraded.
abdelaziz@abdelaziz-VirtualBox:~/try$ lex lex.l
abdelaziz@abdelaziz-VirtualBox:~/try$ lex lex.l
abdelaziz@abdelaziz-VirtualBox:~/try$ gcc lex.yy.c -o lexer -ll
abdelaziz@abdelaziz-VirtualBox:~/try$ ./lexer < test_program
```

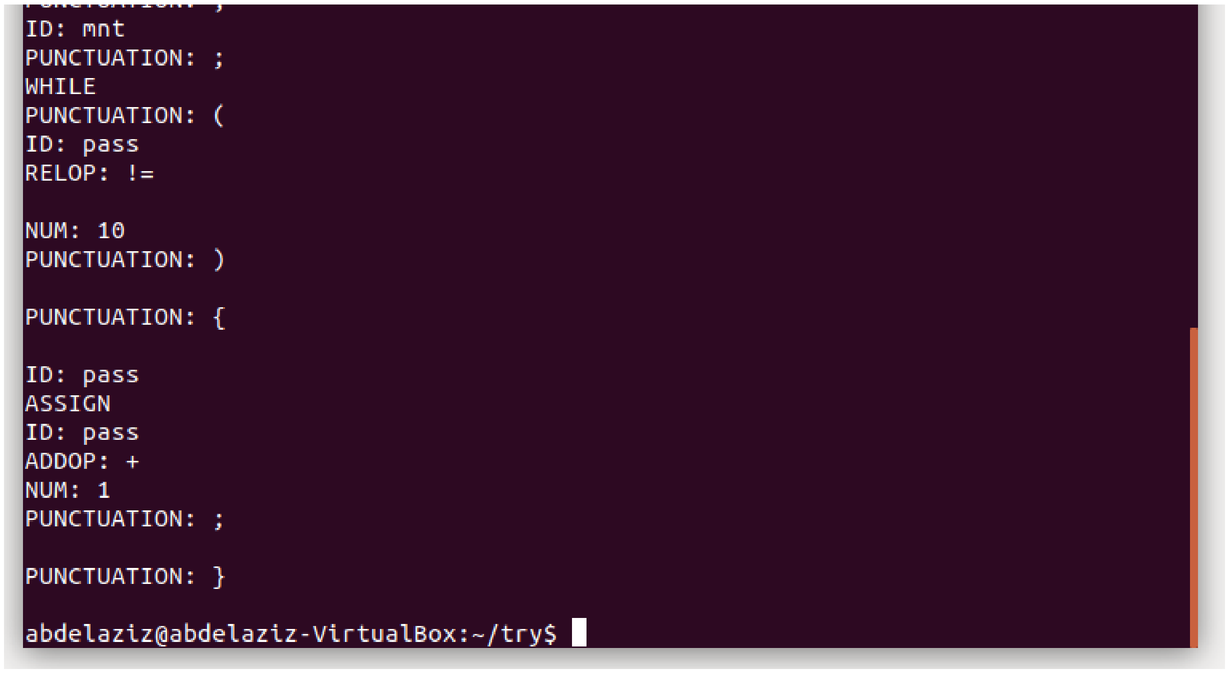


```
abdelaziz@abdelaziz-VirtualBox:~/try$ ./lexer < test_program
INT
ID: sum
PUNCTUATION: ,
ID: count
PUNCTUATION: ,
ID: pass
PUNCTUATION: ,
ID: mnt
PUNCTUATION: ;
WHILE
PUNCTUATION: (
ID: pass
RELOP: !=

NUM: 10
PUNCTUATION: )

PUNCTUATION: {

ID: pass
ASSIGN
ID: pass
ADDOP: +
```



```
ID: mnt
PUNCTUATION: ;
WHILE
PUNCTUATION: (
ID: pass
RELOP: !=

NUM: 10
PUNCTUATION: )

PUNCTUATION: {

ID: pass
ASSIGN
ID: pass
ADDOP: +
NUM: 1
PUNCTUATION: ;

PUNCTUATION: }

abdelaziz@abdelaziz-VirtualBox:~/try$
```

<u>Conclusion:</u>

Lex/Flex is a valuable tool for automating the generation of lexical analyzers, simplifying the process of recognizing and categorizing tokens in a programming language. The provided Lex specification efficiently captures the lexical structure of the language, and the generated lexer can be further integrated into a compiler or used for various analysis tasks.

<u>References:</u>

https://www.geeksforgeeks.org/flex-fast-lexical-analyzer-generator/