Alexandria University CS321
Faculty of Engineering
Compilers
Computer and Systems Engineering Dept.
Third Year

Phase1: Lexical Analyzer Generator
Bonus Report

Names	ID
Peter Atef	19
Radwa Adel	23
Aliaa Othman	41
Mark Philip	49

What is Flex ?

Flex is an open source program designed to automatically and quickly generate scanners, also known as tokenizers, which recognize lexical patterns in text. Flex is an acronym that stands for "fast lexical analyzer generator. " It is a free alternative to Lex, the standard lexical analyzer generator in Unix-based systems. Flex was originally written in the C programming language by Vern Paxson in 1987.

Detailed description of the required steps :

1- install Flex using the following two commands (ubuntu)

```
sudo apt-get update
sudo apt-get install flex
```

2- for compiling a lex program :

```
- write the lex program in a file and save it as file.l
  (where file is the     name of the file).
- open the terminal and navigate to the directory where you
  have saved the  file.l
- type - lex file.l
- then type - cc lex.yy.c -lfl
- then type - ./a.out
```

screenshots for using the tool :

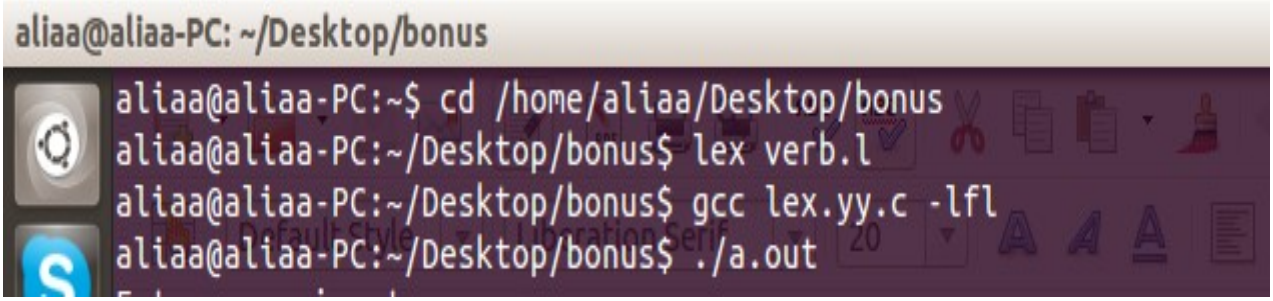
1-file.l

```
letter  [a-z]|[A-Z]
digit   [0-9]
digits  [0-9]+
%%
boolean                                                    printf("boolean\n");
int                                                        printf("int\n");
float                                                      printf("float\n");
if                                                         printf("if\n");
else                                                       printf("else\n");
while                                                      printf("while\n");

{letter}({letter}|{digit})*                                printf("id\n");
{digit}+|{digit}+"."{digits}+("\n"|E{digits}+)             printf("num\n");
==|!=|>|>=|<|<=                                            printf("relop\n");
=                                                          printf("assign\n");
"+"|-                                                      printf("addop\n");
"*"|"/"                                                    printf("mulop\n");
";"                                                        printf(";\n");
","                                                        printf(",\n");
"("                                                        printf("(\n");
")"                                                        printf(")\n");
"{"                                                        printf("{\n");
"}"                                                        printf("}\n");
"]"                                                        printf("]\n");
%%

main()
{
 printf("Enter your input:\n");
 yylex();
}
```

2-compliling the file and generating output for test cases:



aliaa@aliaa-PC: ~/Desktop/bonus

```
Enter your input:
int sum , count , pass ,

mnt; while (pass != 10)
{

pass = pass + 1 ;

}int
 id
 ,
 id
 ,
 id
 ,

id
;
 while
 (
id
 relop
 num
)

{


id
 assign
 id
 addop
 num
 ;



}
```

a.out

lex.yy.c

verb.l

aliaa@aliaa-PC: ~/Desktop/bonus

```
aliaa@aliaa-PC:~$ cd /home/aliaa/Desktop/bonus
aliaa@aliaa-PC:~/Desktop/bonus$ lex verb.l
aliaa@aliaa-PC:~/Desktop/bonus$ gcc lex.yy.c -lfl
aliaa@aliaa-PC:~/Desktop/bonus$ ./a.out
Enter your input:
222.33E55
num

<>aliaa
relop
relop
id

boolean for int /*
boolean
 id
 int
 mulop
mulop

for(int i =0 ; i<5 ; i++){}
id
(
int
 id
 assign
num
 ;
 id
relop
num
 ;
 id
addop
addop
)
{
}
```