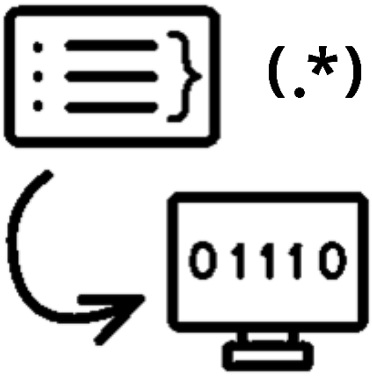
Atma Ram Sanatan Dharma College
University of Delhi



System Programming – Lex

Practical File for Paper Code 32347501

Submitted By
Sudipto Ghosh
College Roll No. 19/78003
BSc (Hons) Computer Science

Submitted To
Ms Manisha Bagri
Department of Computer Science

INDEX

PRACTICAL 1

Objective

Write a Lex program to count the number of vowels and consonants from text file.

Code

```
%{
  int vowels = 0;
  int consonants = 0;
%}
%%
A|E|I|O|U|a|e|i|o|u { vowels++; };
[A-Za-z] { consonants++; };
[ |\n|\t] { ; }
. { ; }
%%
int main()
{
  yylex();
  printf("%i vowels, %i consonants\n", vowels, consonants);
  return 0;
}

int yywrap()
{
  return 1;
}
```

Output

```
$ .\a
System Programming
4 vowels, 13 consonants
```

PRACTICAL 2

Objective

Write a Lex program to identify a word entered by user is a verb or not.

Code

```
%%
is      |
am      |
are     |
were    |
was     |
be      |
being   |
been    |
do      |
does    |
did     |
will    |
would   |
should  |
can     |
could   |
has     |
have    |
had     |
go        { printf("%s is a verb\n", yytext); }
[A-Za-z]+ { printf("%s is not a verb\n", yytext); }
[ |\t|\n] { ; }
. { ; }
%%
int main()
{
  yylex();
  return 0;
}

int yywrap()
{
  return 1;
}
```

Output

```
$ .\a
System does
System is not a verb
does is a verb
```

PRACTICAL 3

Objective

Write a Lex program to count the number of integers, floats, identifiers, operators, and comments in a C program.

Code

```
%{
  int integers = 0;
  int floats = 0;
  int identifiers = 0;
  int operators = 0;
  int comments = 0;
%}
%%
[#].* { printf("%s <- preprocessor directive\n", yytext); } // prepro
cessor directives
[ |\n|\t] { ; } // whitespaces
[,|;|"("|")"|"{"|"}"|"\["|"\]"] { ; } // brackets, delimiters
"//".* { comments++; printf("%s <- comment\n", yytext); } // single l
ine comments
[0-9]+ { integers++; printf("%s <- integer\n", yytext); } // integers
[0-9]+("."[0-
9]+) { floats++; printf("%s <- float\n", yytext); } // floats
void|int|main|char|for|while|continue|switch|case|break|if|else|retur
n|true|false { printf("%s <- keyword\n", yytext); } // keywords
"<="|">="|"!="|"=="|"<"|">"|"&"|"|"|"^"|"<<"|">>"|"~"|"&&"|"||"|"!"|"
++"|"--"|"="|"+"|"-
“|"*"|"/"|"%" { operators++; printf("%s <- operator\n", yytext); } //
 operators
['](([^\\\']|\\.)?['] { ; } // characters
["](([^\\\"]|\\.)*["] { ; } // strings
[a-zA-Z_]+[a-zA-Z0-
9_]* { identifiers++; printf("%s <- identifier\n", yytext); } // iden
tifiers
%%
int main() {
  yyin = fopen("text.c", "r");
  yylex();
  printf("\n");
  printf("number of integers: %d\n", integers);
  printf("number of floats: %d\n", floats);
  printf("number of identifiers: %d\n", identifiers);
  printf("number of operators: %d\n", operators);
  printf("number of comments: %d", comments);
  return 0;
}
```

```c
int yywrap() {
  return 1;
}
```

Input

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void loop()
{
  int i, j = 5; // okay
  for (int i = 0; i < j; i++)
  {
    printf("%d. Hi\n", (i + 1));
  }
}

int main()
{
  double a = -3.14;
  char at = '@', tab = '\t';

  loop();

  return 0;
}

// sudipto
```

Output

```
$ .\a
#include <stdio.h> ← preprocessor directive
#include <stdlib.h> ← preprocessor directive
#include <string.h> ← preprocessor directive
void ← keyword
loop ← identifier
int ← keyword
i ← identifier
j ← identifier
= ← operator
5 ← integer
// okay ← comment
for ← keyword
int ← keyword
i ← identifier
= ← operator
```

```
0 ← integer
i ← identifier
< ← operator
j ← identifier
i ← identifier
++ ← operator
printf ← identifier
i ← identifier
+ ← operator
1 ← integer
int ← keyword
main ← keyword
double ← identifier
a ← identifier
= ← operator
- ← operator
3.14 ← float
char ← keyword
at ← identifier
= ← operator
tab ← identifier
= ← operator
loop ← identifier
return ← keyword
0 ← integer
// sudipto ← comment

number of integers: 4
number of floats: 1
number of identifiers: 14
number of operators: 9
number of comments: 2
```

PRACTICAL 4

Objective

Write a Lex program to count the number of words, characters, blank spaces, and lines in a text file.

Code

```
%{
  int words = 0;
  int lines = 0;
  int spaces = 0;
  int characters = 0;
%}
%%
[^ \t\n,\.:;]+ { words++; characters += yyleng; }
[\n] { lines++; characters += yyleng; }
[ |\t] { spaces++; characters += yyleng; }
. { characters++; }
%%
int main() {
  yyin = fopen("text.txt", "r");
  yylex();
  printf("number of words: %d\n", words);
  printf("number of blank spaces: %d\n", spaces);
  printf("number of lines: %d\n", lines);
  printf("number of characters: %d\n", characters);
  return 0;
}

int yywrap() {
  return 1;
}
```

Input

```
Sudipto Ghosh
College Roll Number 78003
```

Output

```
$ .\a
number of words: 6
number of blank spaces: 4
number of lines: 2
number of characters: 40
```

PRACTICAL 5

Objective

Write a Lex program that implements the Caesar cipher.

Code

```
%{
  int rot = 0;
%}
%%
[A-Z] { fprintf(yyout, "%c", (yytext[0] - 'A' + rot) % 26 + 'A'); }
[a-z] { fprintf(yyout, "%c", (yytext[0] - 'a' + rot) % 26 + 'a'); }
. { fprintf(yyout, "%s", yytext); }
%%
int main(void) {
  printf("Enter Key (ROT): ");
  scanf("%d", &rot);
  yyin = fopen("input.txt", "r");
  yyout = fopen("output.txt", "w");
  yylex();
  fclose(yyin);
  fclose(yyout);
  return 0;
}

int yywrap() {
  return 1;
}
```

Output

```
$ cat .\input.txt
zyu{yf_fy4fu_e00g_13}
xyz
XYZ
$ .\a
Enter Key (ROT): 13
$ cat .\output.txt
mlh{ls_sl4sh_r00t_13}
klm
KLM
```

PRACTICAL 6

Objective

Write a program in Lex to recognize a valid arithmetic expression.

Code

```
%{
  #include <stdio.h>
  int brackets = 0,
      operators = 0,
      numbersOridentifiers = 0,
      flag = 0;
%}
%%
[a-zA-Z_]+[a-zA-Z0-9_]* { numbersOridentifiers++; }
-?[0-9]+("."[0-9]+)? { numbersOridentifiers++; }
[+|\-|*|/|=|\^|%] { operators++; }
"(" { brackets++; }
")" { brackets--; }
";" { flag = 1; }
.|\n { ; }
%%
int main() {
  yylex();
  if (
    (operators + 1) == numbersOridentifiers
      && brackets == 0 && flag == 0
  ) {
    printf("Valid Expression\n");
  } else {
    printf("Invalid Expression\n");
  }
  return 0;
}

int yywrap() {
  return 1;
}
```

Output

```
$ .\a
Enter Arithmetic Expression: a + b ) * 2
Invalid Expression
$ .\a
Enter Arithmetic Expression: (1 + 2) * (3 - 4)
Valid Expression
```

PRACTICAL 7

Objective

Write a Lex program that finds the longest word in the input.

Code

```
%{
  int length = 0;
  char *word = NULL;
%}
%%
[a-zA-Z]+ {
  if (yyleng > length) {
    length = yyleng;
    word = yytext;
  }
}
[ |\n|\t] { ; }
. { ; }
%%
int main(void) {
  yyin = fopen("input.txt", "r");
  yylex();
  fclose(yyin);
  printf("Longest Word: %.*s\n", length, word);
  printf("Length of Longest Word: %d\n", length);
  return 0;
}

int yywrap() {
  return 1;
}
```

Output

```
$ cat .\input.txt
words or characters
pneumonoultramicroscopicsilicovolcanokoniosis
$ .\a
Longest Word: pneumonoultramicroscopicsilicovolcanokoniosis
Length of Longest Word: 45
```