

System Programming and Compiler Construction

VI Semester (Computer)

Academic Year: 22-23

Experiment No 5

Aim : Study of Lexical analyzer tool -Flex/ Lex

Source Code:

I: Program for counting number of vowels and consonant

```
%{
#include <stdio.h>
int vow=0,con=0;
%}
%%
[aeiouAEIOU] {vow++;}
[a-zA-Z] {con++;}
%%
int yywrap(){
    return 1;
}
int main(){
    printf("Enter the string:");
    yylex();
    printf("Vowles: %d \nConsonants: %d\n",vow,con);
    return 0;
}
```

Output:

#lex alphalex.l

#gcc lex.yy.c

#!/a.out

This Lex program counts the number of vowels and consonants in given text.

Enter the text and terminate it with CTRL-d.

```
universe@acer6:~/Downloads$ lex prac4a.l
universe@acer6:~/Downloads$ cc lex.yy.c -ll
universe@acer6:~/Downloads$ ./a.out
Enter the string:Hi Kris

Vowles: 2
Consonants: 4
```

2: Lexical Analyzer for Number Classification

```
%{
#include <stdio.h>
int pos_int = 0, pos_frac = 0;
int neg_int = 0, neg_frac = 0;
%}
%%
[+]?[0-9]* {pos_int++;}
[+]?[0-9]*\.[0-9]+ {pos_frac++;}
[-]?[0-9]* {neg_int++;}
[-]?[0-9]*\.[0-9]+ {neg_frac++;}
%%

int yywrap(){
    return 1;
}

int main(){
    printf("Enter the number:\n");
    yylex();
    printf("Positive Integer: %d\n", pos_int);
    printf("Positive Fracion: %d\n", pos_frac);
    printf("Negative Integer: %d\n", neg_int);
    printf("Negative Fraction: %d\n", neg_frac);
    return 0;
}
```

Output:

```
universe@acer6:~/Downloads$ lex prac4b.l
universe@acer6:~/Downloads$ cc lex.yy.c -ll
universe@acer6:~/Downloads$ ./a.out
Enter the number:
25

-12.5

-69

420

2.9

Positive Integer: 2
Positive Fracion: 1
Negative Integer: 1
Negative Fraction: 1
```

3: Lexical Analyzer for Counting Operators and Operands in a String

```
%{
#include <stdio.h>
int operator=0,operand=0;
}%
%%
[+\-\\*\//\%^] {operator++;}
[a-zA-Z0-9]+ {operand++;}
%%
int yywrap(){
    return 1;
}
int main(){
    printf("Enter the string:");
    yylex();
    printf("Operators: %d \nOperands: %d\n",operator,operand);
    return 0;
}
```

Output:

```
universe@acer6:~/Downloads$ lex prac4c.l
universe@acer6:~/Downloads$ cc lex.yy.c -ll
universe@acer6:~/Downloads$ ./a.out
Enter the string:a=b+c*d-69
=
Operators: 3
Operands: 5
universe@acer6:~/Downloads$
```

4: Lex program to count characters, words, spaces, and new lines in a string.

```
%{
#include <stdio.h>
int chars = 0, word = 0, newLine = 0, space = 0;
%}

%%

[a-zA-Z]{1} {chars++;}
[a-zA-Z]{2,} {word++;}
[ ] {space++;}
[\n] {newLine++;}

%%

int yywrap(){
return 1;
}

int main()
{
printf("Enter String: ");
yylex();
printf("Number of characters is: %d\n", chars);
printf("Number of words is: %d\n", word);
printf("Number of spaces is: %d\n", space);
printf("Number of new line characters is: %d\n", newLine);
}
```

Output:

```
universe@acer6:~/Downloads$ lex prac4d.l
universe@acer6:~/Downloads$ cc lex.yy.c -ll
universe@acer6:~/Downloads$ ./a.out
Enter String: printf("Good"/n"Morning")
(""/"")Number of characters is: 1
Number of words is: 3
Number of spaces is: 0
Number of new line characters is: 1
universe@acer6:~/Downloads$ █
```

5: Lex program to count characters, words, spaces, and new lines in a string.

```
%{
#include <stdio.h>
int keyword=0;
int operator=0;
int special=0;
int identifier=0;
}%
%%
[int||char||float||return||if||else||do||while] {keyword++;}
[=+|-|*|\/|^] {operator++;}
[a-zA-Z0-9]+ {identifier++;}
[%/&/$/;] {special++;}

%%
int yywrap(){
    return 1;
}
int main(){
    printf("Enter the code:");
    yylex();
    printf("Keyword: %d \nOperators: %d \nIdentifier: %d \nSpecial
symbols: %d",keyword,operator,identifier,special);
    return 0;
}
```

Output:

```
Special symbols: 0universe@acer6:~/Downloads$ lex prac4e.l
universe@acer6:~/Downloads$ cc lex.yy.c -ll
universe@acer6:~/Downloads$ ./a.out
Enter the code:int a = b + 6c

Keyword: 1
Operators: 2
Identifier: 3
Special symbols: 0universe@acer6:~/Downloads$
```

Aim : Study of Parser generator tool – Yacc

first.l:

```
%{
    /* Definition section*/
#include "y.tab.h"
extern yyval;
}%

%%

[0-9]+ { yyval = atoi(yytext); return NUMBER; }
[a-zA-Z]+ { return ID; }
[\t]+ ;
\n { return 0; }
. { return yytext[0]; }
%%

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

first.y:

```
%{
    /* Definition section */
#include<stdio.h>
}%

%token NUMBER ID
// setting the precedence
// and associativity of operators
%left '+' '-'
%left '*' '/'

/* Rule Section */
%%

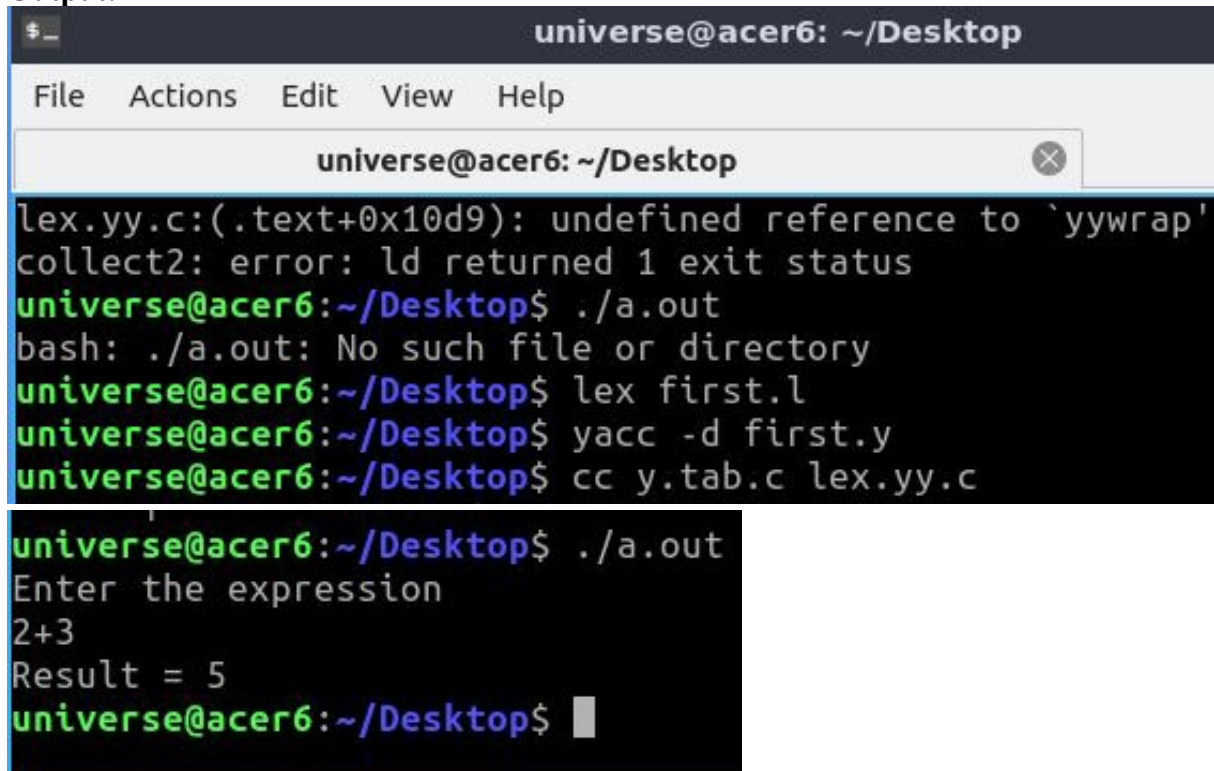
E : T { printf("Result = %d\n", $$); return 0; }

T : T '+' T { $$ = $1 + $3; }
  | T '-' T { $$ = $1 - $3; }
  | T '*' T { $$ = $1 * $3; }
  | T '/' T { $$ = $1 / $3; }
  | NUMBER { $$ = $1; }
%%
```

```
int main()
{
printf("Enter the expression\n");
yyvsparse();
}

/* For printing error messages */
int yyerror(char* s)
{
printf("\nExpression is invalid\n");
}
```

Output:



```
universe@acer6: ~/Desktop
File Actions Edit View Help
universe@acer6: ~/Desktop
lex.yy.c:(.text+0x10d9): undefined reference to `yywrap'
collect2: error: ld returned 1 exit status
universe@acer6:~/Desktop$ ./a.out
bash: ./a.out: No such file or directory
universe@acer6:~/Desktop$ lex first.l
universe@acer6:~/Desktop$ yacc -d first.y
universe@acer6:~/Desktop$ cc y.tab.c lex.yy.c
universe@acer6:~/Desktop$ ./a.out
Enter the expression
2+3
Result = 5
universe@acer6:~/Desktop$
```