

**Compiler Construction Lab Practicals
Document**

Submitted by

Poorva Pohekar, 22070521108

B. Tech Computer Science and Engineering

Under the Guidance of

Dr.Snehlata K Wankhade

Symbiosis Institute of Technology, Nagpur.



**SYMBIOSIS
INSTITUTE OF TECHNOLOGY, NAGPUR**

Wathoda, Nagpur
2025

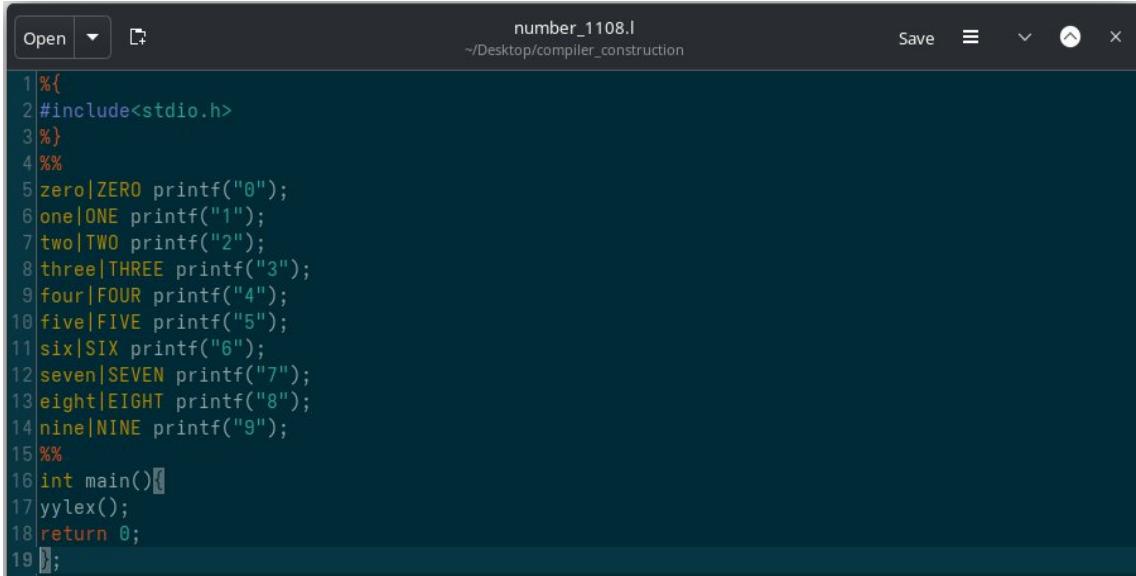
PRACTICAL -01

Aim: Theory Assignment for writing details about Lex and Yacc compiler

Code:

```
%{
#include<stdio.h>
%}
%%
zero|ZERO printf("0");
one|ONE printf("1");
two|TWO printf("2");
three|THREE printf("3");
four|FOUR printf("4");
five|FIVE printf("5");
six|SIX printf("6");
seven|SEVEN printf("7");
eight|EIGHT printf("8");
nine|NINE printf("9");
%%
int main(){
yylex();
return 0;
};
```

Gedit Code(number_1108.l):



```
number_1108.l
~/Desktop/compiler_construction

1 %{
2 #include<stdio.h>
3 %
4 %%
5 zero|ZERO printf("0");
6 one|ONE printf("1");
7 two|TWO printf("2");
8 three|THREE printf("3");
9 four|FOUR printf("4");
10 five|FIVE printf("5");
11 six|SIX printf("6");
12 seven|SEVEN printf("7");
13 eight|EIGHT printf("8");
14 nine|NINE printf("9");
15 %%
16 int main(){
17 yylex();
18 return 0;
19 };
```

Output:

```
(base) [avroop@arch compiler_construction]$ lex number_1108.l
(base) [avroop@arch compiler_construction]$ cc lex.yy.c -lfl
(base) [avroop@arch compiler_construction]$ ./a.out
zero
0
one
1
two
2
three
3
four
4
five
5
SIX
6
SEVEN
7
EIGHT
EIGHT
EIGHT
8
nine
9
```

Practical-02

Aim: Count the numbers of comments, keywords, identifiers, word and space line from users input.

Code:

```
%{
#include <stdio.h>
int lc = 0;
int sc = 0;
int cmnt = 0;
int kw = 0;
int id = 0;
int wc = 0;
int tc = 0;
%}

%%

/*([^\n]*+[^\n]*)*\n+{/ cmnt++; }

'int'|'float'|'char'|'if'|'else'|'while'|'for'|'return'|'void'|'do'|'main'|'break'|'continue' {
    kw++;
    wc++;
}
\t {tc++; }

[a-zA-Z_][a-zA-Z0-9_]* {
    id++;
    wc++;
}
[ \t]+ { sc += yyleng; }
\n { lc++; }

%
int main() {
    printf("Enter a sentence: (press ctrl+d to end)");
    yylex();
    printf("\n--- Result ---\n");
    printf("Lines: %d\n", lc);
    printf("Spaces: %d\n", sc);
    printf("Comments: %d\n", cmnt);
    printf("Keywords: %d\n", kw);
    printf("Identifiers: %d\n", id);
    printf("Words: %d\n", wc);
    printf("Tab count: %d\n", tc);
    return 0;
}
```

```

}

int yywrap() {
    return 1;
}

```

Gedit Code(count_1108.l):

```

1 /*
2 #include <stdio.h>
3
4 int lc = 0;
5 int sc = 0;
6 int cmnt = 0;
7 int kw = 0;
8 int id = 0;
9 int wc = 0;
10 int tc = 0;
11 */
12
13 %%
14
15 /*([^\n]*[^\n]+)*\n*/ { cmnt++; }
16
17 "int"|"float"|"char"|"if"|"else"|"while"|"for"|"return"|"void"|"do"|"main"|"break"|"continue"| {
18     kw++;
19     wo++;
20 }
21 \t {tc++; }
22
23 [a-zA-Z_][a-zA-Z0-9_]* {
24     id++;
25     wc++;
26 }
27 [\t]+ { sc += yyleng; }
28 \n { lc++; }
29
30 %%
31
32 int main() {
33     printf("Enter a sentence: (press ctrl+d to end)");
34     yylex();
35     printf("\n--- Result ---\n");
36     printf("Lines: %d\n", lc);
37     printf("Spaces: %d\n", sc);
38     printf("Comments: %d\n", cmnt);
39     printf("Keywords: %d\n", kw);
40     printf("Identifiers: %d\n", id);
41     printf("Words: %d\n", wc);
42     printf("Tab count: %d\n", tc);
43     return 0;
44 }
45
46 int yywrap() {
47     return 1;
48 }
49
50

```

Output:

```
(base) [avroop@arch prac-02]$ lex count_1108.l
(base) [avroop@arch prac-02]$ cc lex.yy.c
(base) [avroop@arch prac-02]$ ./a.out
Enter a sentence: (press ctrl+d to end)
hi i am poorva pohekar
im learning lex compiler
/*vvcryebry*/
--- Result ---
Lines: 4
Spaces: 7
Comments: 1
Keywords: 0
Identifiers: 9
Words: 9
Tab count: 0
(base) [avroop@arch prac-02]$ █
```

Prcatical-03

Aim: To write lex program that counts number of words starting with the letter 'a' or 'A' from given input text.

Code:

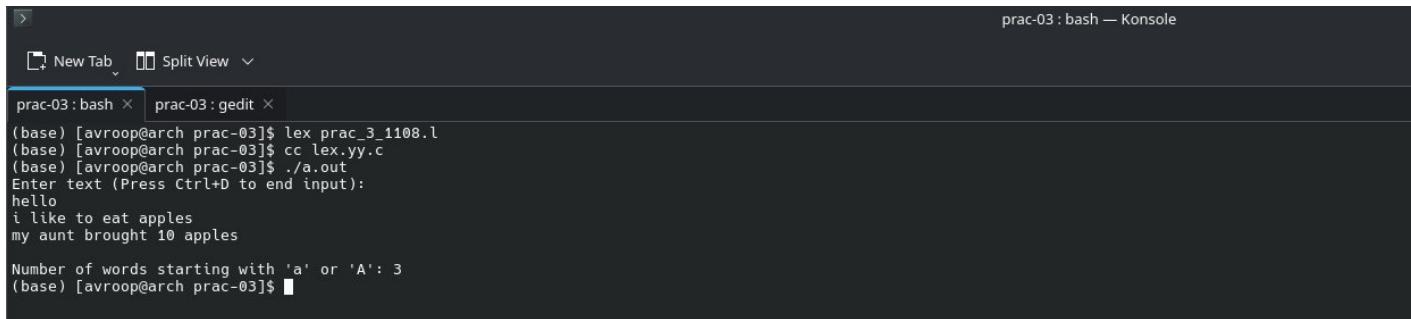
```
%{
#include <stdio.h>
int count = 0;
%}
%%
[aA][a-zA-Z]* { count++; }      // Count word starting with 'a' or 'A'
[^a-zA-Z]+   { /* Skip non-words */ }
[a-zA-Z]+   { /* Skip other words */ }
%%
int yywrap() {
    return 1;
}
int main() {
    printf("Enter text (Press Ctrl+D to end input):\n");
    yylex();
    printf("\nNumber of words starting with 'a' or 'A': %d\n", count);
    return 0;
}
```

Gedit Code(prac_3_1108.l):



```
1 %{  
2 #include <stdio.h>  
3 int count = 0;  
4 %}  
5  
6 %%  
7 [aA][a-zA-Z]* { count++; } // Count word starting with 'a' or 'A'  
8 [^a-zA-Z]+ { /* Skip non-words */ }  
9 [a-zA-Z]+ { /* Skip other words */ }  
10  
11 %%  
12  
13  
14 int yywrap() {  
15     return 1;  
16 }  
17  
18 int main() {  
19     printf("Enter text (Press Ctrl+D to end input):\n");  
20     yylex();  
21     printf("\nNumber of words starting with 'a' or 'A': %d\n", count);  
22     return 0;  
23 }
```

Output:



```
prac-03 : bash — Konsole  
[prac-03 : bash] prac-03 : gedit  
(base) [avroop@arch prac-03]$ lex prac_3_1108.l  
(base) [avroop@arch prac-03]$ cc lex.yy.c  
(base) [avroop@arch prac-03]$ ./a.out  
Enter text (Press Ctrl+D to end input):  
hello  
i like to eat apples  
my aunt brought 10 apples  
Number of words starting with 'a' or 'A': 3  
(base) [avroop@arch prac-03]$
```

Practical-04

Aim: Conversion of lowercase to uppercase and vice-versa.

Code:

```
%{  
#include <stdio.h>  
%}  
  
%%  
[A-Z] { putchar(yytext[0] + 32); } // Uppercase to lowercase  
[a-z] { putchar(yytext[0] - 32); } // Lowercase to uppercase
```

```

.\n      { ECHO; }           // Print all other characters as-is
%%
int yywrap() {
    return 1;
}
int main() {
    printf("Enter text (Press Ctrl+D to end input):\n");
    yylex();
    return 0;
}

```

Gedit Code(upp_low_1108.l):

```

1 %{
2 #include <stdio.h>
3 %
4
5 %%
6
7 [A-Z]      { putchar(yytext[0] + 32); } // Uppercase to lowercase
8 [a-z]      { putchar(yytext[0] - 32); } // Lowercase to uppercase
9 .|\n      { ECHO; }           // Print all other characters as-is
10
11 %%
12
13 int yywrap() {
14     return 1;
15 }
16
17 int main() {
18     printf("Enter text (Press Ctrl+D to end input):\n");
19     yylex();
20     return 0;
21 }
22

```

Output:

```

prac-04 : a.out — Konsole
prac-04 : a.out × | prac-04 : gedit ×
(base) [avroop@arch prac-04]$ lex upp_low_1108.l
(base) [avroop@arch prac-04]$ cc lex.yy.c
(base) [avroop@arch prac-04]$ ./a.out
Enter text (Press Ctrl+D to end input):
hii im doing LEX PROGRAM
HIII IM DOING lex program

```

Practical-05

Aim: Conversion of Decimal number to it's Hexadecimal equivalent from user input.

Code:

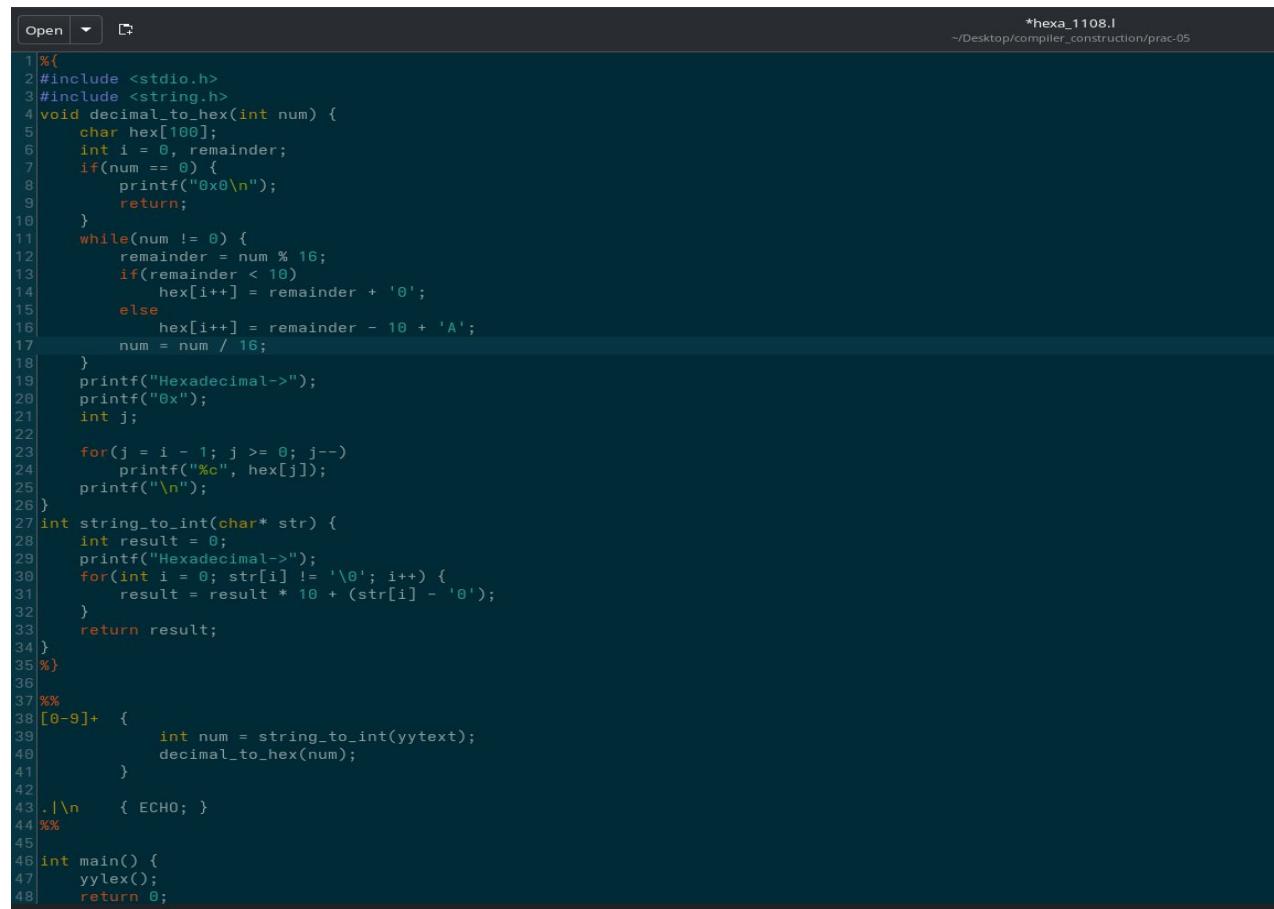
```
%{
#include <stdio.h>
#include <string.h>
void decimal_to_hex(int num) {
    char hex[100];
    int i = 0, remainder;
    if(num == 0) {
        printf("0x0\n");
        return;
    }
    while(num != 0) {
        remainder = num % 16;
        if(remainder < 10)
            hex[i++] = remainder + '0';
        else
            hex[i++] = remainder - 10 + 'A';
        num = num / 16;
    }
    printf("0x");
    int j;
    for(j = i - 1; j >= 0; j--)
        printf("%c", hex[j]);
    printf("\n");
}
int string_to_int(char* str) {
    int result = 0;
    for(int i = 0; str[i] != '\0'; i++) {
        result = result * 10 + (str[i] - '0');
    }
    return result;
}
%%}
```

```
[0-9]+ {
    int num = string_to_int(yytext);
    decimal_to_hex(num);
}
```

```
.|\n { ECHO; }
%%
```

```
int main() {
    yylex();
    return 0;
}
int yywrap() {
    return 1;
}
```

Gedit Code(hexa_1108.l):



```
*hexa_1108.l
~/Desktop/compiler_construction/prac-05

1 %}
2 #include <stdio.h>
3 #include <string.h>
4 void decimal_to_hex(int num) {
5     char hex[100];
6     int i = 0, remainder;
7     if(num == 0) {
8         printf("0x0\n");
9         return;
10    }
11    while(num != 0) {
12        remainder = num % 16;
13        if(remainder < 10)
14            hex[i++] = remainder + '0';
15        else
16            hex[i++] = remainder - 10 + 'A';
17        num = num / 16;
18    }
19    printf("Hexadecimal->");
20    printf("0x");
21    int j;
22
23    for(j = i - 1; j >= 0; j--)
24        printf("%c", hex[j]);
25    printf("\n");
26}
27 int string_to_int(char* str) {
28     int result = 0;
29     printf("Hexadecimal->");
30     for(int i = 0; str[i] != '\0'; i++) {
31         result = result * 10 + (str[i] - '0');
32     }
33     return result;
34}
35 %}
36
37 %%
38 [0-9]+ {
39     int num = string_to_int(yytext);
40     decimal_to_hex(num);
41 }
42
43 .|\n { ECHO; }
44 %%
45
46 int main() {
47     yylex();
48     return 0;
```

Output:

```
prac-05 : a.out — Konsole
prac-05 : gedit ×  prac-05 : a.out ×
(base) [avroop@arch prac-05]$ lex hexa_1108.l
(base) [avroop@arch prac-05]$ cc lex.yy.c
(base) [avroop@arch prac-05]$ ./a.out
126
Hexadecimal->0x7E
12
Hexadecimal->0xC
396789
Hexadecimal->0x60DF5
```

Practical-06

Aim: To write a Lex program that identifies and counts lines ending with the substring "com" from the input text and prints the matched lines.

Code:

```
%{
#include <stdio.h>
int ln=0;
%}
%%
.*com$ { ln++; printf("Matched line: %d %s",ln, yytext); }
.*
%%
int main() {
    printf("Enter text here: \n");
    yylex();
    return 0;
}
int yywrap() {
    return 1;
}
```

Gedit Code(com_1108.l):

```
Open ▾ com_1108.l
~/Desktop/compiler_construction/prac-06

1 %{
2 #include <stdio.h>
3 int ln=0;
4 %}
5
6 /**
7 .*com$ { ln++; printf("Matched line: %d %s",ln, yytext); }
8 .*
9 /**
10
11 int main() {
12     printf("Enter text here: \n");
13     yylex();
14     return 0;
15 }
16 int yywrap() {
17     return 1;
18 }
```

Output:

```
prac-06 : a.out — Konsole
New Tab Split View
New Tab (Ctrl+Shift+T) prac-06 : a.out × prac-06 : gedit ×
(base) [avroop@arch prac-06]$ lex com_1108.l
(base) [avroop@arch prac-06]$ cc lex.yy.c
(base) [avroop@arch prac-06]$ ./a.out
Enter text here:
hi

i am poorva

my id is pogvtwvy@gmail.com
Matched line: 1 my id is pogvtwvy@gmail.com
```