

# CD LAB 1

Name - Rithvik Rajesh Matta

SRN - PES2UG23CS485

Class - 6H

## Counts.l

```
CD > PE1-Students >  counts.l
 1  %{
 2  |     int nchar, nword, nline;
 3  %}
 4  %%
 5  \n      { nline++; nchar++; }
 6  [^ \t\n]+ { nword++, nchar += yyleng; }
 7  .       { nchar++; }
 8  %%
 9  int yywrap()
10 {
11     return(1);
12 }
13 int main(int argc, char *argv[])
14 {
15     yyin = fopen(argv[1], "r");
16     yylex();
17     printf("Number of Characters : %d\nNumber of Words: %d\nNumber of lines: %d\n", nchar, nword, nline);
18     return 0;
19 }
```

Valid input

```
CD > PE1-Students >  input.txt
 1  Hello world
 2  This is a test
 3
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ gcc lex.yy.c -o counts
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./counts < input.txt
Number of Characters : 27
Number of Words: 6
Number of lines: 2
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

Invalid input

```
CD > PE1-Students > ⌂ input.txt
1  Generate code (Ctrl+I), or select a language (Ctrl+K M). Start typing to dismiss
   or don't show this again.

pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./counts < input.txt
Number of lines: 2
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ Number of Characters : 0
Number of Words: 0
Number of lines: 0
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

## Example1.l

```
CD > PE1-Students > ⌂ example1.l
1  %}
2  #include<stdio.h>
3  %
4  %%
5  abb      printf("1");
6  aba      printf("2");
7  a       printf("3");
8  %%
9  int yywrap()
10 {
11     return(1);
12 }
13 int main(int argc, char *argv[])
14 {
15     yyin = fopen(argv[1], "r");
16     yylex();
17     fclose(yyin);
18     return 0;
19 }
```

Valid input

```
CD > PE1-Students > input.txt
1 abb
2 aba
3 a
4
3bc
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./example1 < input.txt
1
2
3
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

Invalid input

```
CD > PE1-Students > input.txt
1 b
2 ab
3 ba
4 bb
5 abc
6
3
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./example1 < input.txt
b
3b
b3
bb
3bc
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

## Example2.l

```
CD > PE1-Students > 例 example2.l
1  %
2  #include<stdio.h>
3  %
4  %
5  a*b      printf("1");
6  (a|b)*b  printf("2");
7  c*       printf("3");
8  %
9  int yywrap()
10 {
11     return(1);
12 }
13 int main(int argc, char *argv[])
14 {
15     yyin = fopen(argv[1], "r");
16     yylex();
17     fclose(yyin);
18     return 0;
19 }
```

Valid input

```
CD > PE1-Students > 例 input.txt
1  b
2  ab
3  aaab
4  bb
5  abb
6  aab
7
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ gcc lex.yy.c -o example2
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./example2 < input.txt
1
1
1
2
2
1
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

## Invalid input

```
CD > PE1-Students >   input.txt
1 a
2 ba
3 ac
4 bc
5 abc
6
2
1
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./example2 < input.txt
a
1a
a3
13
13
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

## Example3.l

```
CD > PE1-Students >  ≡ example3.l
1  %
2  #include<stdio.h>
3  %
4  %
5  aa          printf("1");
6  b?a+b?      printf("2");
7  b?a*b?      printf("3");
8  %
9  int yywrap()
10 {
11     return(1);
12 }
13 int main(int argc, char *argv[])
14 {
15     yyin = fopen(argv[1], "r");
16     yylex();
17     fclose(yyin);
18     return 0;
19 }
```

### Valid input

```
CD > PE1-Students > input.txt
1 aa
2 a
3 aaa
4 ba
5 aab
6 bab
7 ab
8 b
9 bb
10
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./example3 < input.txt
1
2
2
2
2
2
3
3
3
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

### Invalid input

```
CD > PE1-Students > input.txt
1 c
2 ac
3 bc
4 bbb
5 aba
6
2
2
3
3
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./example3 < input.txt
c
2c
3c
33
22
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

## Identifiers.l

```
CD > PE1-Students > ☰ identifiers.l
1 digit      [0-9]
2 letter     [_A-Za-z]
3 %}
4 #include<stdio.h>
5 %}
6 %%
7 {letter}({letter}|{digit})*    printf("Valid Identifier : %s\n",yytext);
8 .
9 \
10 %
11 int yywrap()
12 {
13     return(1);
14 }
15 int main(int argc, char *argv[])
16 {
17     yyin = fopen(argv[1], "r");
18     yylex();
19     fclose(yyin);
20     return 0;
21 }
```

## Valid input

```
CD > PE1-Students > ☰ input.txt
1 a
2 abc
3 A1
4 _var
5 temp123
6 my_var_2
7 ABC_def
8
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ flex identifiers.l
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ gcc lex.yy.c -o identifiers
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./identifiers < input.txt
Valid Identifier : a
Valid Identifier : abc
Valid Identifier : A1
Valid Identifier : _var
Valid Identifier : temp123
Valid Identifier : my_var_2
Valid Identifier : ABC_def
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

## invalid input

The screenshot shows a terminal window with the following content:

```
CD > PE1-Students > input.txt
1 labc
2 2_var
3 9
4 @var
5 #temp
6 123abc
7 a labc temp123 _x 9var myVar2 @t
8
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./identifiers < input.txt
Valid Identifier : temp123
Valid Identifier : my_var_2
Valid Identifier : ABC_def
Valid Identifier : abc
Valid Identifier : _var
Valid Identifier : var
Valid Identifier : temp
Valid Identifier : abc
Valid Identifier : a
Valid Identifier : abc
Valid Identifier : temp123
Valid Identifier : _X
Valid Identifier : var
Valid Identifier : myVar2
Valid Identifier : test
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

## Keywords\_and\_identifiers.l

```
CD > PE1-Students >  keywords_and_identifiers.l
 1  digit      [0-9]
 2  letter     [_A-Za-z]
 3  %{
 4  #include<stdio.h>
 5  %}
 6  %%
 7  auto|double|if|static|break|else|int|struct|case|enum|long|switch|char|extern|near|typedef|const|float|continue|register|union|unsi
 8  {letter}({letter}|{digit})*    printf("Valid Identifier : %s\n",yytext);
 9  . ;
10  \n ;
11  %%
12  int yywrap()
13  {
14  |    return(1);
15  }
16  int main(int argc, char *argv[])
17  [
18  |    yyin = fopen(argv[1], "r");
19  |    yylex();
20  |    fclose(yyin);
21  |    return 0;
22  ]
```

### Valid input

The screenshot shows a VS Code interface with three tabs: 'Walkthrough: Setup VS Code', 'input.txt', and 'example1.l'. The terminal window displays the output of the lexical analyzer 'Kai' running on the file 'input.txt'. The output shows tokens being identified: 'int', 'float', 'double', 'char', 'if', 'else', 'while', '\_var', 'temp123', 'my\_var\_2', 'count', and 'value9'. Each token is labeled as either a 'Keyword' or a 'Valid Identifier'.

```
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./Kai < input.txt
Keyword :      int
Keyword :      float
Keyword :      double
Keyword :      char
Keyword :      if
Keyword :      else
Keyword :      while
Valid Identifier : _var
Valid Identifier : temp123
Valid Identifier : my_var_2
Valid Identifier : count
Valid Identifier : value9
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

## Invalid input

```
CD > PE1-Students > input.txt
1 int main count labc float _var wh
2 char signed unsigned temp123
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students\$ ./Kai < input.txt

Identifier	Type
Valid Identifier : value9	Value
Valid Identifier : main	Keyword
Valid Identifier : count	Identifier
Valid Identifier : abc	Identifier
Keyword : float	Keyword
Valid Identifier : _var	Identifier
Keyword : while	Keyword
Valid Identifier : x	Identifier
Valid Identifier : value9	Value
Valid Identifier : test	Identifier
Keyword : char	Keyword
Keyword : signed	Identifier
Keyword : unsigned	Identifier
Valid Identifier : temp123	Identifier

pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students\$

## Keywords.l

```
CD > PE1-Students > keywords.l
1  %{
2  #include<stdio.h>
3  %
4  %%
5  auto|double|if|static|break|else|int|struct|case|enum|long|switch|char|extern|near|typedef|const|float|continue|regis
6  .
7  \
8  %%
9  int yywrap()
10 {
11     return(1);
12 }
13 int main(int argc, char *argv[])
14 {
15     yyin = fopen(argv[1], "r");
16     yylex();
17     fclose(yyin);
18     return 0;
19 }
```

## valid

```
CD > PE1-Students > input.txt
1 auto
2 double
3 if
4 static
5 break
6 else
7 int
8 struct
9 case
10 enum
11
```

pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students\$ ./Key < input.txt

Identifier	Type
Keyword : auto	Keyword
Keyword : double	Identifier
Keyword : if	Keyword
Keyword : static	Identifier
Keyword : break	Identifier
Keyword : else	Identifier
Keyword : int	Identifier
Keyword : struct	Identifier
Keyword : case	Identifier
Keyword : enum	Identifier

pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students\$

invalid

The screenshot shows a terminal window with two panes. The left pane displays the contents of 'input.txt' which contains a series of C keywords on separate lines: main, return, printf, for, sizeof, volatile, short, and bool. The right pane shows the command 'pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students\$ ./Key < input.txt' being run. The output shows the word 'Keyword' followed by a tab and the word 'int', indicating a syntax error.

```
CD > PE1-Students > input.txt
1 main
2 return
3 printf
4 for
5 sizeof
6 volatile
7 short
8 bool
9
```

```
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./Key < input.txt
Keyword :      int
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

## Line\_numbers.l

The screenshot shows a terminal window with two panes. The left pane displays the contents of 'line\_numbers.l', a Lex script. It includes definitions for yylineno, a rule for newlines, a yywrap function returning 1, and a main function opening 'input.txt', calling yylex, and closing the file. The right pane shows the command 'pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students\$ ./line < input.txt' being run. The output lists the three lines from 'input.txt' (Hello world, This is line two, This is line three) each preceded by its line number (1, 2, 3).

```
CD > PE1-Students > line_numbers.l
1 %{           int yylineno;
2 %}           %
3 %%           %
4 (.*)(\n|$)   printf("%d\t%s", yylineno++, yytext);
5 %%           %
6 int yywrap()
7 {
8     return(1);
9 }
10 int main(int argc, char *argv[])
11 {
12     yyin = fopen(argv[1], "r");
13     yylex();
14     fclose(yyin);
15 }
```

```
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./line < input.txt
1 Hello world
2 This is line two
3 This is line three
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

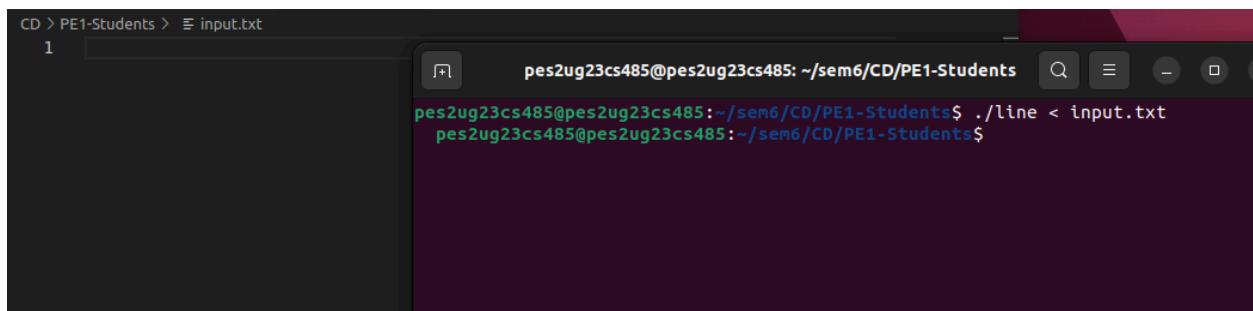
Valid

The screenshot shows a terminal window with two panes. The left pane displays the contents of 'input.txt' which contains three lines of text: 'Hello world', 'This is line two', and 'This is line three'. The right pane shows the command 'pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students\$ ./line < input.txt' being run. The output lists the three lines from 'input.txt' (Hello world, This is line two, This is line three) each preceded by its line number (1, 2, 3).

```
CD > PE1-Students > input.txt
1 Hello world
2 This is line two
3 This is line three
4
```

```
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./line < input.txt
1 Hello world
2 This is line two
3 This is line three
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```

invalid



The screenshot shows a terminal window with a dark background and light-colored text. At the top left, it says "CD > PE1-Students > input.txt". In the center, there is a command prompt: "pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students\$". Below the prompt, the user has typed the command ". /line < input.txt". The terminal then displays an error message: "pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students\$". This indicates that the command was executed successfully, but the output was empty or did not produce the expected results.

```
CD > PE1-Students > input.txt
1
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$ ./line < input.txt
pes2ug23cs485@pes2ug23cs485:~/sem6/CD/PE1-Students$
```