



VIT[®]

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Assessment 2

Bhoomi Vijay
23BCI0051

13 August 2025

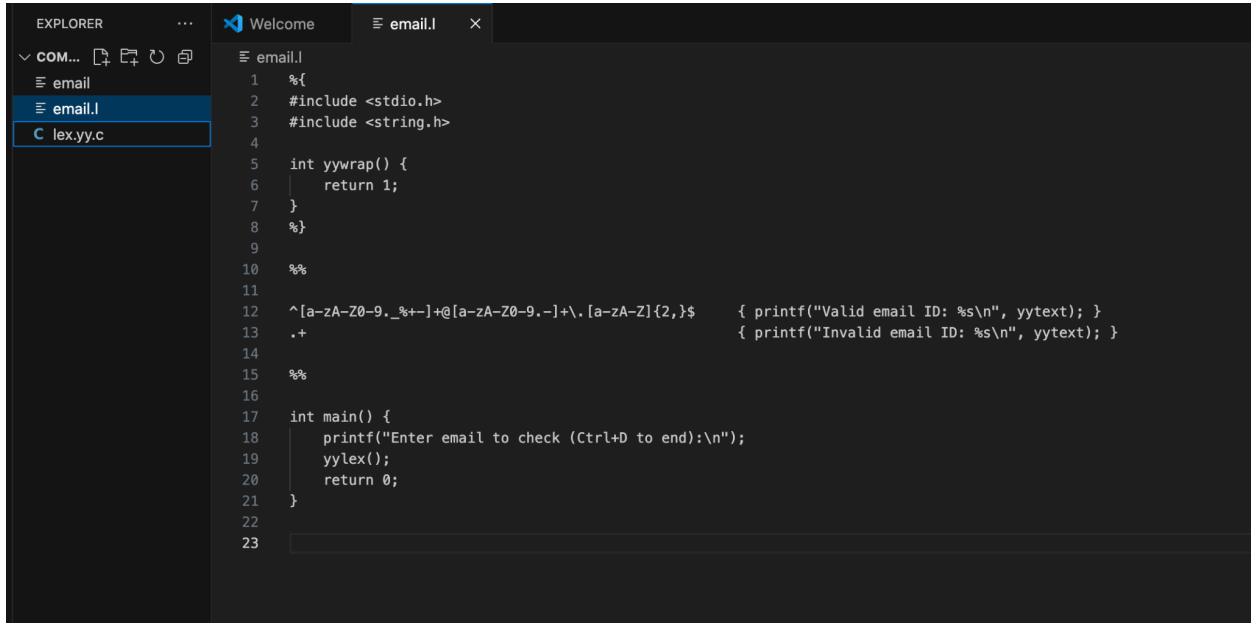
Lab Slot- L47 +L48

Q1. Write lex code to check valid email or not.

Code:

```
%{  
#include <stdio.h>  
#include <string.h>  
  
int yywrap() {  
    return 1;  
}  
%}  
  
%%  
  
^[a-zA-Z0-9._%+~]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ { printf("Valid email ID: %s\n", yytext); }  
.+ { printf("Invalid email ID: %s\n", yytext); }  
  
%%  
  
int main() {  
    printf("Enter email to check (Ctrl+D to end):\n");
```

```
yylex();  
return 0;  
}
```



```
1  %{  
2  #include <stdio.h>  
3  #include <string.h>  
4  
5  int yywrap() {  
6      return 1;  
7  }  
8  %}  
9  
10 %*  
11  
12 ^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ { printf("Valid email ID: %s\n", yytext); }  
13 .+ { printf("Invalid email ID: %s\n", yytext); }  
14  
15 %*  
16  
17 int main() {  
18     printf("Enter email to check (Ctrl+D to end):\n");  
19     yylex();  
20     return 0;  
21 }  
22  
23
```

Input & Output:

```
bhoomi@Bhoomis-MacBook-Air compiler % flex email.l  
gcc lex.yy.c -o email -ll
```

```
bhoomi@Bhoomis-MacBook-Air compiler % ./email
```

Enter email to check (Ctrl+D to end):

bhoomi

Invalid email ID: bhoomi

bhoomi@gmail.com

Valid email ID: bhoomi@gmail.com

```

bhoomi@Bhoomis-MacBook-Air compiler % flex email.l

bhoomi@Bhoomis-MacBook-Air compiler % gcc lex.yy.c -o email -I/opt/homebrew/opt/flex/include -L/opt/homebrew/opt/flex/lib -lfl

bhoomi@Bhoomis-MacBook-Air compiler % ./email

Enter email to check (Ctrl+D to end):
test@example.com
Valid email ID: test@example.com

^C
bhoomi@Bhoomis-MacBook-Air compiler % ./email

Enter email to check (Ctrl+D to end):
bad-em@
Invalid email ID: bad-em@

```

Q2. Write a lex code to check whether the mobile number is 10 digits.

Code:

```

%{
#include <stdio.h>

int yywrap() {
    return 1;
}
%}

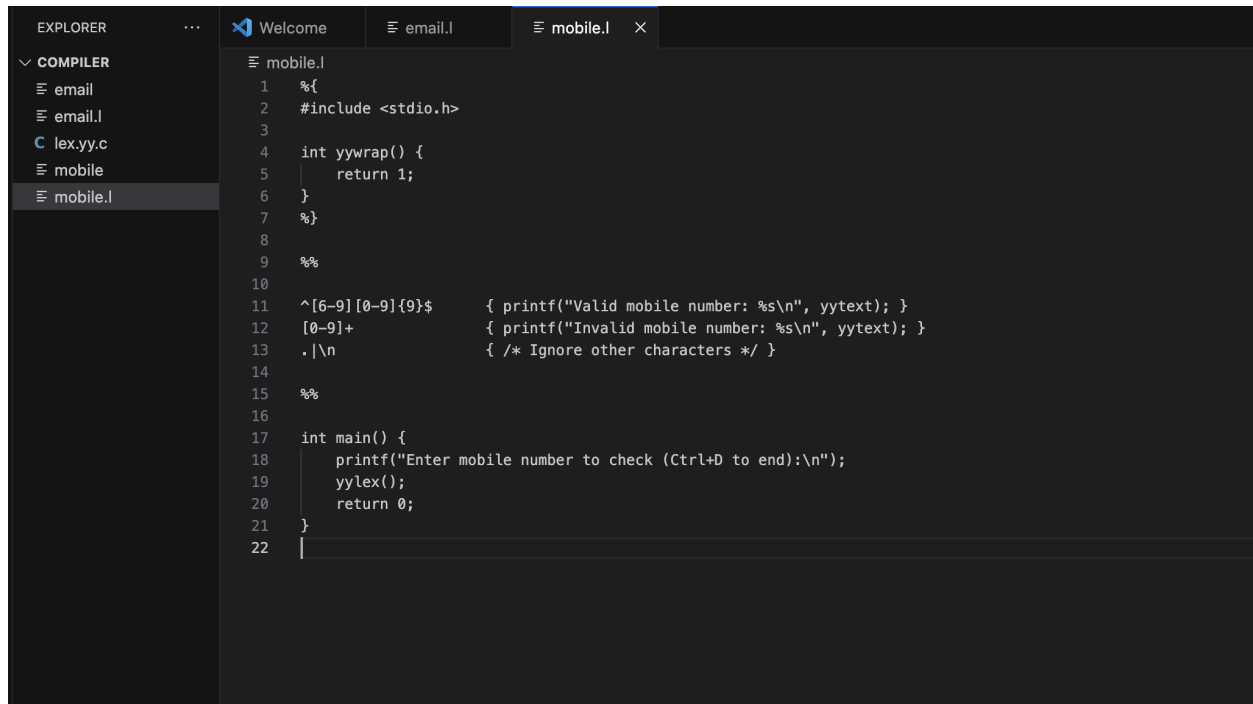
%%

^[6-9][0-9]{9}$    { printf("Valid mobile number: %s\n", yytext); }
[0-9]+             { printf("Invalid mobile number: %s\n", yytext); }
.\n                { /* Ignore other characters */ }

%%

int main() {
    printf("Enter mobile number to check (Ctrl+D to end):\n");
    yylex();
    return 0;
}

```



```
1  %{
2  #include <stdio.h>
3
4  int yywrap() {
5      return 1;
6  }
7  %}
8
9  %%
10
11 ^[6-9][0-9]{9}$      { printf("Valid mobile number: %s\n", yytext); }
12 [0-9]+              { printf("Invalid mobile number: %s\n", yytext); }
13 .|\n                { /* Ignore other characters */ }
14
15 %%
16
17 int main() {
18     printf("Enter mobile number to check (Ctrl+D to end):\n");
19     yylex();
20     return 0;
21 }
22 |
```

Input & Output:

bhoomi@Bhoomis-MacBook-Air compiler % flex mobile.l

gcc lex.yy.c -o mobile -ll

bhoomi@Bhoomis-MacBook-Air compiler % ./mobile

Enter mobile number to check (Ctrl+D to end):

6543212

Invalid mobile number: 6543212

9821346452

Valid mobile number: 9821346452

987654

Invalid mobile number: 987654

```

bhoomi@Bhoomis-MacBook-Air compiler % flex mobile.l

bhoomi@Bhoomis-MacBook-Air compiler % gcc lex.yy.c -o mobile -I/opt/homebrew/opt/flex/include -L/opt/homebrew/opt/flex/lib -lfl

bhoomi@Bhoomis-MacBook-Air compiler % ./mobile

Enter mobile number to check (Ctrl+D to end):
9123548710
Valid mobile number: 9123548710
^C
bhoomi@Bhoomis-MacBook-Air compiler % ./mobile

Enter mobile number to check (Ctrl+D to end):
64257
Invalid mobile number: 64257

```

Q3. Write a lex code to check whether a URL is valid or not.

Code:

```

%option noyywrap

%{
    #include <stdio.h>
%}

url www\.[a-zA-Z0-9-]+\(\.[a-zA-Z]{2,}\)+

%%

{url}    {printf("valid url");}

.+       {printf("invalid url");}

%%

int main()

```

```

{

    yylex();

    return 0;

}

```

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer lists files: check, check.l, email, email.l, lex.yy.c, lexer, lexical.l, mobile, and mobile.l. The code editor shows the content of check.l, which is a flex input file. It includes `<stdio.h>` and defines two rules: `url` and `.*`. The `url` rule matches a valid URL and prints "valid url". The `.*` rule matches any invalid input and prints "invalid url". The `main` function calls `yylex()` and returns 0.

```

1  %option noyywrap
2  %{
3      #include <stdio.h>
4  %}
5  url www\.[a-zA-Z0-9-]+\(\.[a-zA-Z]{2,}\)+
6  %%
7  {url}      {printf("valid url");}
8  .+        {printf("invalid url");}
9  %%
10 int main()
11 {
12     yylex();
13     return 0;
14 }

```

Input & Output:

bhoomi@Bhoomis-MacBook-Air compiler % flex check.l

gcc lex.yy.c -o check -ll

./check

www.google.com

www.my-site.org

hello

valid url

valid url

invalid url

```

bhoomi@Bhoomis-MacBook-Air compiler % flex check.1

bhoomi@Bhoomis-MacBook-Air compiler % gcc lex.yy.c -o check -I/opt/homebrew/opt/flex/include -L/opt/homebrew/opt/flex/lib -lfl

bhoomi@Bhoomis-MacBook-Air compiler % ./check
www.vit.ac.in
valid url
bhoomi@3
invalid url

```

Q4. Lexical Analyser

Code:

```

%{
#include <stdio.h>
int line_num = 1;
int yywrap() { return 1; }
}%

%%

"int"|"float"|"char"|"if"|"else"|"while"|"return" { printf("Keyword: %s\n", yytext); }
"=="|"!="|"<="|">="|"<"|>" { printf("Relational Operator: %s\n", yytext); }
"="|"+"|"-"|"*"|"/" { printf("Operator: %s\n", yytext); }
"(" | ")" | "{" | "}" | ";" { printf("Symbol: %s\n", yytext); }
[0-9]+ { printf("Number: %s\n", yytext); }
[a-zA-Z_][a-zA-Z0-9_]* { printf("Identifier: %s\n", yytext); }
\".*\" { printf("String Literal: %s\n", yytext); }
\n { line_num++; }
[ \t]+ { /* Ignore whitespace */ }
. { printf("Unknown token: %s\n", yytext); }

%%

int main() {
    printf("Enter code (Ctrl+D to end):\n");
    yylex();
    return 0;
}

```

```

1  %{
2  #include <stdio.h>
3  int line_num = 1;
4  int yywrap() { return 1; }
5  %{
6
7  %%
8
9  "int"|"float"|"char"|"if"|"else"|"while"|"return" { printf("Keyword: %s\n", yytext); }
10 "="|"!="|"<="|">="|"<"|>" { printf("Relational Operator: %s\n", yytext); }
11 "+"|"-"|"*"|"/" { printf("Operator: %s\n", yytext); }
12 "("|")"|"{"|"}"|";" { printf("Symbol: %s\n", yytext); }
13 "[0-9]+" { printf("Number: %s\n", yytext); }
14 "[a-zA-Z][a-zA-Z0-9_]*" { printf("Identifier: %s\n", yytext); }
15 "\\\".*\\" { printf("String Literal: %s\n", yytext); }
16 "\\n" { line_num++; }
17 "[ \t]+" { /* Ignore whitespace */ }
18 "." { printf("Unknown token: %s\n", yytext); }
19
20 %%
21
22 int main() {
23     printf("Enter code (Ctrl+D to end):\n");
24     yylex();
25     return 0;
26 }
27

```

Output:

bhoomi@Bhoomis-MacBook-Air compiler % flex lexical.l

gcc lex.yy.c -o lexical -ll

bhoomi@Bhoomis-MacBook-Air compiler % ./lexical

Enter code (Ctrl+D to end):

```

int main() {
    int x = 10;
    float y = 20.5;
Keyword: int
Identifier: main
Number: (
Unknown token: )
Unknown token: {
Keyword: int
Identifier: x
Operator: =
Number: 10
Unknown token: ;
Keyword: float
Identifier: y
Operator: =
Number: 20
Unknown token: .

```


Number: 5
Unknown token: ;
x = x + y;
if (x > 15) {
printf("Hello World");
return 0;
}
Identifier: printf
Number: (
String Literal: "Hello World"
Unknown token:)
Unknown token: ;
Unknown token: }
Keyword: return
Number: 0
Unknown token: ;
Unknown token: }

○ bhoomi@Bhoomis-MacBook-Air compiler % ./lexical

Enter code (Ctrl+D to end):

```
int main() {
    int x = 10;
    float y = 20.5;
    Keyword: int
    Identifier: main
    Number: (
    Unknown token: )
    Unknown token: {
    Keyword: int
    Identifier: x
    Operator: =
    Number: 10
    Unknown token: ;
    Keyword: float
    Identifier: y
    Operator: =
    Number: 20
    Unknown token: .
    Number: 5
    Unknown token: ;
    x = x + y;
    if (x > 15) {
        printf("HeIdentifier: x
    Operator: =
    Identifier: x
    Operator: +
    Identifier: y
    Unknown token: ;
    Keyword: if
    Number: (
    Identifier: x
    Relational Operator: >
    Number: 15
    Unknown token: )
    Unknown token: {
    llo World");
    }
    return 0;
}
Identifier: printf
Number: (
String Literal: "Hello World"
Unknown token: )
Unknown token: ;
Unknown token: }
Keyword: return
Number: 0
Unknown token: ;
Unknown token: }
```

Q5. Write a LEX code to count single line comment and multiline comments in a file.

```
Code:%{
#include <stdio.h>

int single_count = 0;
int multi_count = 0;
%}

%%
"/".*      { single_count++; }
"/*"([^\]|\"+[^/] )*\"/*\" { multi_count++; }
.          { /* ignore other characters */ }
\n         { /* ignore newlines */ }
%%

int main(int argc, char *argv[]) {
    if (argc > 1) {
        FILE *file = fopen(argv[1], "r");
        if (!file) {
            perror("Error opening file");
            return 1;
        }
        yyin = file;
    }

    yylex();

    printf("Single-line comments: %d\n", single_count);
    printf("Multi-line comments: %d\n", multi_count);

    return 0;
}

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

```

≡ comment_counter.l
1  %{
2  #include <stdio.h>
3
4  int single_count = 0;
5  int multi_count = 0;
6  %}
7
8  %%
9  "//".*          { single_count++; }
10 "/"([^\]|\\*+[^/])*"*/" { multi_count++; }
11 .              { /* ignore other characters */ }
12 \n             { /* ignore newlines */ }
13 %%
14
15 int main(int argc, char *argv[]) {
16     if (argc > 1) {
17         FILE *file = fopen(argv[1], "r");
18         if (!file) {
19             perror("Error opening file");
20             return 1;
21         }
22         yyin = file;
23     }
24
25     yylex();
26
27     printf("Single-line comments: %d\n", single_count);
28     printf("Multi-line comments: %d\n", multi_count);
29
30     return 0;
31 }
32
33 int yywrap(void) {
34     return 1;
35 }
36

```

Input file:

```

#include <stdio.h>

int main() {
    // Single-line comment 1
    // Single-line comment 2

    /* Multi-line comment 1 */

    printf("Hello World\n"); // Single-line comment 3
}

```

```

/* Multi-line comment 2
   still multi-line */

int x = 5;
/* Multi-line comment 3 */
printf("Value: %d\n", x);

// Single-line comment 4

return 0;
}

```

Output:

```

bhoomi@Bhoomis-MacBook-Air compiler % flex comment_counter.l
gcc lex.yy.c -o comment_counter -ll
./comment_counter input.c

```

Single-line comments: 4

Multi-line comments: 3

```

bhoomi@Bhoomis-MacBook-Air compiler %

```

```

Single-line comments: 4
Multi-line comments: 3
● bhoomi@Bhoomis-MacBook-Air compiler % flex comment_counter.l
gcc lex.yy.c -o comment_counter -ll
./comment_counter input.c

Single-line comments: 4
Multi-line comments: 3
○ bhoomi@Bhoomis-MacBook-Air compiler %

```