

COUSRE: CSA1445 COMPILER DESIGN FOR POLYMORPHIC FUNCTIONS

NAME: P.PANEENDRA – 192321072

LEX PROGRAMS

1) Write a LEX program to identify the capital words from the given input.

Aim:

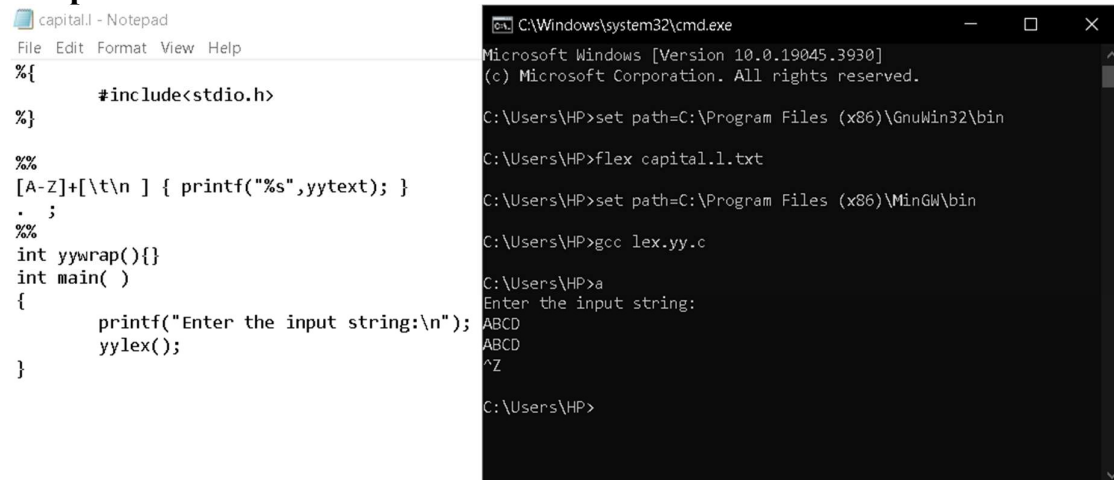
To develop a LEX program that identifies and prints all capital words from the given input.

Code:

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

}%
%%
[A-Z]+[\t\n ] { printf("%s",yytext); }
. ;
%%
Int yywrap(){}
int main( )
{
    printf("Enter the input string:\n");
    yylex();
}
```

Output:



The image shows two windows side-by-side. The left window is a Notepad++ editor titled 'capital.l - Notepad++' with a menu bar (File, Edit, Format, View, Help). It contains the LEX program code shown in the previous block. The right window is a Windows Command Prompt titled 'C:\Windows\system32\cmd.exe'. It shows the following commands and output:

```
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin

C:\Users\HP>flex capital.l.txt

C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin

C:\Users\HP>gcc lex.yy.c

C:\Users\HP>a
Enter the input string:
ABCD
ABCD
^Z

C:\Users\HP>
```

2. Validate Email Address

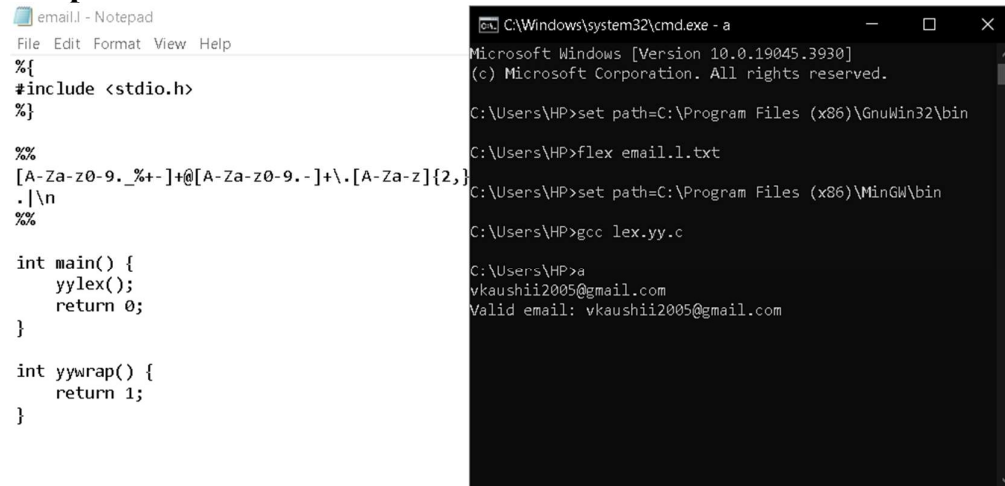
Aim: To implement a LEX program that checks whether a given email address follows a valid format using pattern matching.

Code:

```
%{
%}
%%
[a-zA-Z0-9_]+@[a-z]+\.".com"|.in" { printf("it is valid");}
.+ { printf("it is not valid");}
%%

int yywrap(){}
int main()
{
printf("enter the mail:");
yylex();
}
```

Output:



The image shows two windows side-by-side. The left window is a Notepad editor titled 'email.l - Notepad' with a menu bar (File, Edit, Format, View, Help). It contains the following code:

```
%{
%}
#include <stdio.h>
%}

%%
[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}
.|\\n
%%

int main() {
    yylex();
    return 0;
}

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

The right window is a Windows Command Prompt titled 'C:\Windows\system32\cmd.exe - a'. It shows the following commands and output:

```
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin
C:\Users\HP>flex email.l.txt
C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin
C:\Users\HP>gcc lex.yy.c
C:\Users\HP>a
vkaushii2005@gmail.com
Valid email: vkaushii2005@gmail.com
```

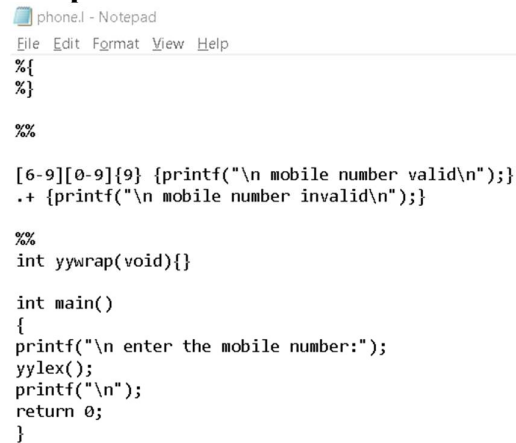
3. Validate Mobile Number

Aim: To create a LEX program that verifies whether a given mobile number is valid based on a specific format (e.g., Indian mobil numbers starting with 7, 8, or 9 and having 10 digits).

Code:

```
%{
#include <stdio.h>
%}
%%
[789][0-9]{9} { printf("Valid mobile number: %s\n", yytext); }
.|n          { /* Ignore other characters */ }
%%
int main() {
    yylex();
    return 0;
}
int yywrap() {
    return 1;
}
```

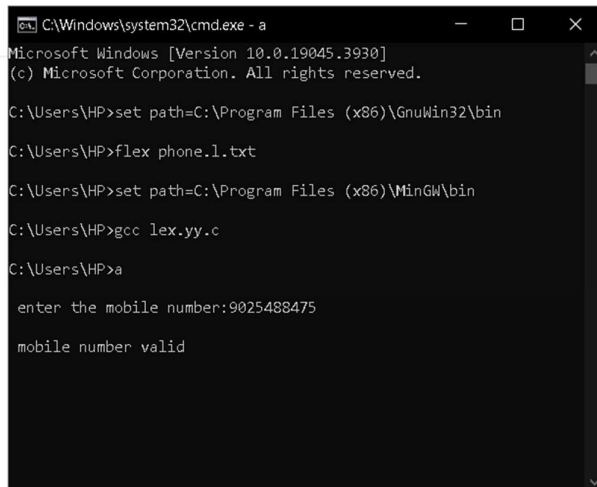
Output:



phone.l - Notepad

```
File Edit Format View Help
%{
%}
%%
[6-9][0-9]{9} {printf("\n mobile number valid\n");}
.+ {printf("\n mobile number invalid\n");}
%%
int yywrap(void){}

int main()
{
printf("\n enter the mobile number:");
yylex();
printf("\n");
return 0;
}
```



```
C:\Windows\system32\cmd.exe - a
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin

C:\Users\HP>flex phone.l.txt

C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin

C:\Users\HP>gcc lex.yy.c

C:\Users\HP>a

enter the mobile number:9025488475

mobile number valid
```

4. Count the Number of Vowels

Aim: To design a LEX program that scans an input sentence and counts the number of vowels (both uppercase and lowercase).

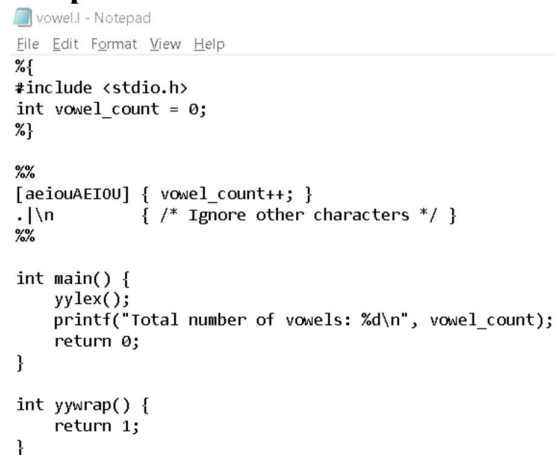
Code:

```
%{
#include <stdio.h>
int vowel_count = 0;
%}
%%
[aeiouAEIOU] { vowel_count++; }
.|\\n      { /* Ignore other characters */ }
%%

int main() {
    yylex();
    printf("Total number of vowels: %d\\n", vowel_count);
    return 0;
}

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

Output:



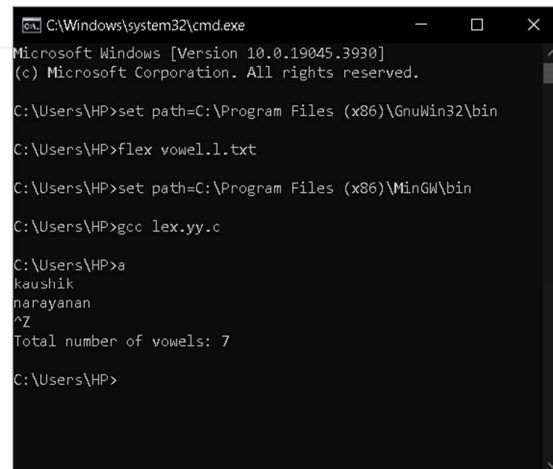
Notepad window titled 'vowel.l - Notepad'. The code is as follows:

```
File Edit Format View Help
%{
#include <stdio.h>
int vowel_count = 0;
%}

%%
[aeiouAEIOU] { vowel_count++; }
.|\\n      { /* Ignore other characters */ }
%%

int main() {
    yylex();
    printf("Total number of vowels: %d\\n", vowel_count);
    return 0;
}

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



Windows Command Prompt window titled 'C:\Windows\system32\cmd.exe'. The output is as follows:

```
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin
C:\Users\HP>flex vowel.l.txt
C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin
C:\Users\HP>gcc lex.yy.c
C:\Users\HP>a
kaushik
narayanan
^Z
Total number of vowels: 7

C:\Users\HP>
```

5. Check if Input is a Digit

Aim: To write a LEX program that determines whether the given input consists of digits and prints an appropriate message.

Code:

```
%{
#include <stdio.h>
%}
%%
[0-9]+ { printf("Input is a digit: %s\n", yytext); }
.|\\n { /* Ignore other characters */ }
%%

int main() {
    yylex();
    return 0;
}

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

Output:



The image shows two windows side-by-side. The left window is a Notepad application titled 'digit1 - Notepad'. It contains the LEX program code shown in the 'Code' section. The right window is a Windows Command Prompt titled 'C:\Windows\system32\cmd.exe - a'. It shows the following commands and output:

```
C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin
C:\Users\HP>flex digit.1.txt
C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin
C:\Users\HP>gcc lex.yy.c
C:\Users\HP>a
123456789
Input is a digit: 123456789
```

6. LEX Program to Count Characters, Words, and Lines in a C File

Aim:

To write a LEX specification file that reads a C program from a .c file and counts the number of characters, words, and lines.

Code:

```
lex
%{
#include <stdio.h>
int char_count = 0, word_count = 0, line_count = 0;
%}
%%
\n { line_count++; char_count++; }
[\t] { char_count++; }
[a-zA-Z0-9_]+ { word_count++; char_count += yyleng; }
. { char_count++; }
%%

int main(int argc, char *argv[]) {
    if (argc > 1) {
        FILE *file = fopen(argv[1], "r");
        if (!file) {
            printf("Error: Cannot open file %s\n", argv[1]);
            return 1;
        }
        yyin = file;
    }
    yylex();
    printf("Characters: %d\nWords: %d\nLines: %d\n", char_count, word_count,
line_count);
    return 0;
}
```

Output:

```
count1 - Notepad
File Edit Format View Help
%{
int nlines,nwords,nchars;
%}

%%
\n {
    nchars++;nlines++;
}

[^ \n\t]+ {nwords++, nchars=nchars+yyleng;}
. {nchars++;}

%%

int yywrap(void) {}
int main()
{
    yylex();
    printf("Lines = %d\nChars=%d\nWords=%d",nlines,nchars,nwords);

    return 0;
}
```

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin

C:\Users\HP>flex count.1.txt

C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin

C:\Users\HP>gcc lex.yy.c

C:\Users\HP>a
#include <stdio.h>
int main() {
    printf("Hello, World!");
    return 0;
}

^Z
Lines = 6
Chars=78
Words=10
C:\Users\HP>
```

7. LEX Program to Print All Constants in a Given C File

Aim:

To write a LEX specification file that prints all numeric and string constants in a given C program.

Code:

```
%{
}%

%%

<INITIAL>[0-9]+ {printf("Integer\n");}
<INITIAL>[0-9]+[.][0-9]+ {printf("Float\n");}
<INITIAL>[A-Za-z0-9_]* {printf("Identifier\n");}
<INITIAL>[^\\n] {printf("Invalid\n");}

%%

int yywrap(){}
int main()
{
printf("Enter String\n");
yylex();
return 0;
}
```

Output:

printcon.l - Notepad

File Edit Format View Help

```
%{
}%
```

```
%%
```

```
<INITIAL>[0-9]+ {printf("Integer\n");}
<INITIAL>[0-9]+[.][0-9]+ {printf("Float\n");}
<INITIAL>[A-Za-z0-9_]* {printf("Identifier\n");}
<INITIAL>[^\\n] {printf("Invalid\n");}
```

```
%%
```

```
%%
```

```
int yywrap(){}

```

```
int main()
{
printf("Enter String\n");
yylex();
return 0;
}
```

```
C:\Windows\system32\cmd.exe - a
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin

C:\Users\HP>flex printcon.l.txt

C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin

C:\Users\HP>gcc lex.yy.c

C:\Users\HP>a
Enter String
int a = 100;
Identifier
Invalid
Identifier
Invalid
Invalid
Invalid
Integer
Invalid

float b = 3.14;
Identifier
Invalid
Identifier
Invalid
Invalid
Invalid
```

8. LEX Program to Count Macros and Header Files in a C Program

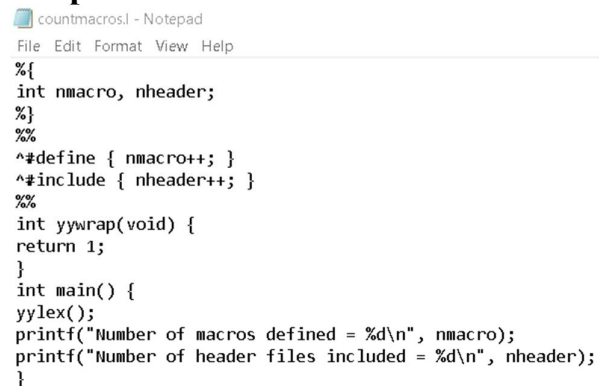
Aim:

To write a LEX specification file that counts the number of macros (#define) and header files (#include) in a C program.

Code:

```
%{
int nmacro, nheader;
}%
%%
^#define { nmacro++; }
^#include { nheader++; }
%%
int yywrap(void) {
return 1;
}
int main() {
yylex();
printf("Number of macros defined = %d\n", nmacro);
printf("Number of header files included = %d\n", nheader);
}
```

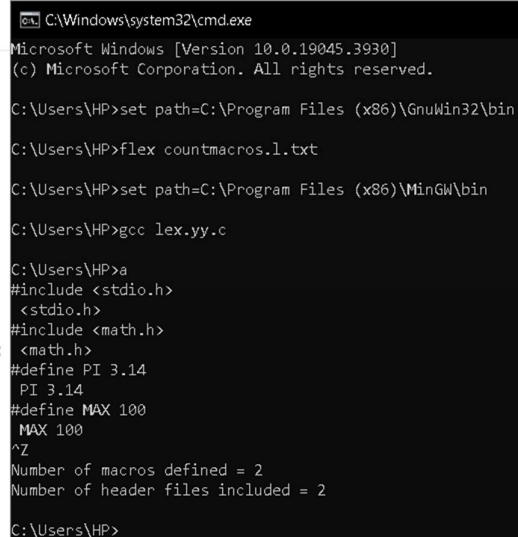
Output:



countmacros.l - Notepad

File Edit Format View Help

```
%{
int nmacro, nheader;
}%
%%
^#define { nmacro++; }
^#include { nheader++; }
%%
int yywrap(void) {
return 1;
}
int main() {
yylex();
printf("Number of macros defined = %d\n", nmacro);
printf("Number of header files included = %d\n", nheader);
}
```



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin

C:\Users\HP>flex countmacros.l.txt

C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin

C:\Users\HP>gcc lex.yy.c

C:\Users\HP>a
#include <stdio.h>
<stdio.h>
#include <math.h>
<math.h>
#define PI 3.14
PI 3.14
#define MAX 100
MAX 100
^Z
Number of macros defined = 2
Number of header files included = 2

C:\Users\HP>
```


9. LEX Program to Print All HTML Tags in an Input File

Aim:

To write a LEX specification file that prints all HTML tags in an input file.

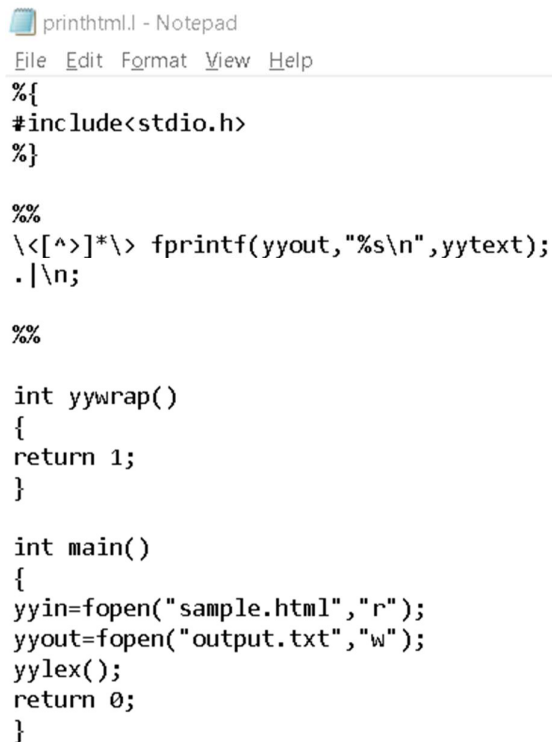
Code:

```
%{
#include <stdio.h>
%}

%%
<[^>]+> { printf("Tag: %s\n", yytext); }
%%

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

Output:



```
printhtml.l - Notepad
File Edit Format View Help

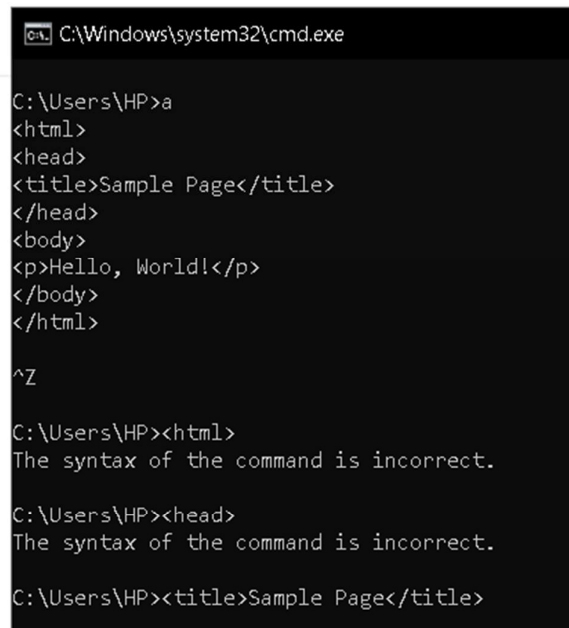
%{
#include<stdio.h>
%}

%%
\[<^>]*\> fprintf(yyout,"%s\n",yytext);
.\|\\n;

%%

int yywrap()
{
return 1;
}

int main()
{
yyin=fopen("sample.html","r");
yyout=fopen("output.txt","w");
yylex();
return 0;
}
```



```
C:\Windows\system32\cmd.exe

C:\Users\HP>a
<html>
<head>
<title>Sample Page</title>
</head>
<body>
<p>Hello, World!</p>
</body>
</html>

^Z

C:\Users\HP><html>
The syntax of the command is incorrect.

C:\Users\HP><head>
The syntax of the command is incorrect.

C:\Users\HP><title>Sample Page</title>
```

10. LEX Program to Add Line Numbers to a C Program

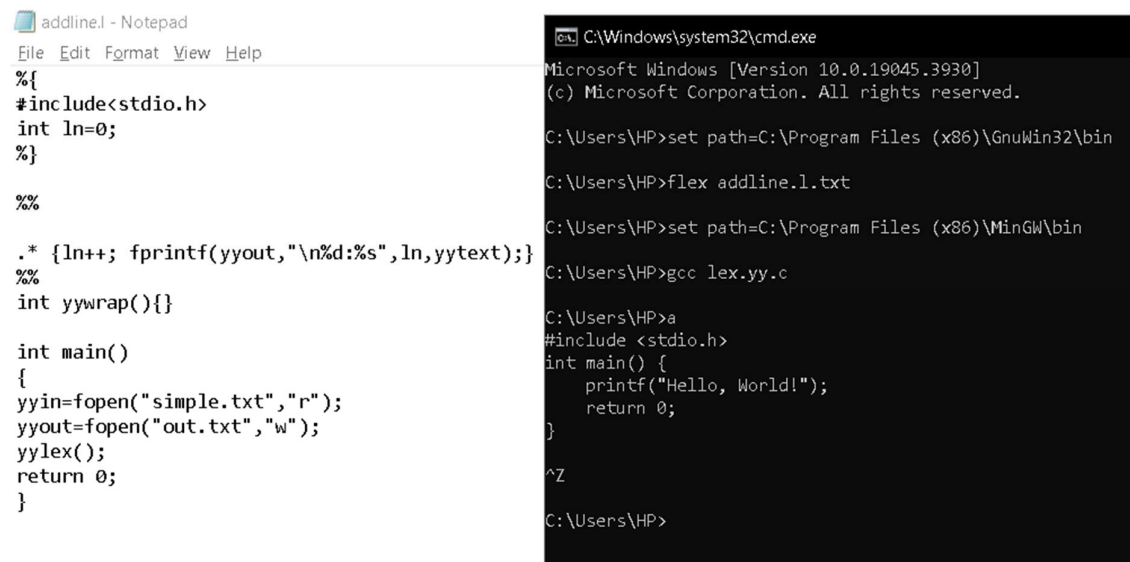
Aim:

To write a LEX specification file that adds line numbers to a C program and displays the modified output.

Code:

```
%{
#include <stdio.h>
int line_no = 1;
%}
%%
^.* { printf("%d %s\n", line_no++, yytext); }
%%
int main() {
    yylex();
    return 0;
}
```

Output:



```
addline.l - Notepad
File Edit Format View Help
%{
#include<stdio.h>
int ln=0;
%}
%%
.* {ln++; fprintf(yyout, "\n%d:%s", ln, yytext);}
%%
int yywrap(){}

int main()
{
yyin=fopen("simple.txt", "r");
yyout=fopen("out.txt", "w");
yylex();
return 0;
}

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin

C:\Users\HP>flex addline.l.txt

C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin
C:\Users\HP>gcc lex.yy.c

C:\Users\HP>a
#include <stdio.h>
int main() {
    printf("Hello, World!");
    return 0;
}

^Z

C:\Users\HP>
```

11. LEX Program to Count and Remove Comments in a C File

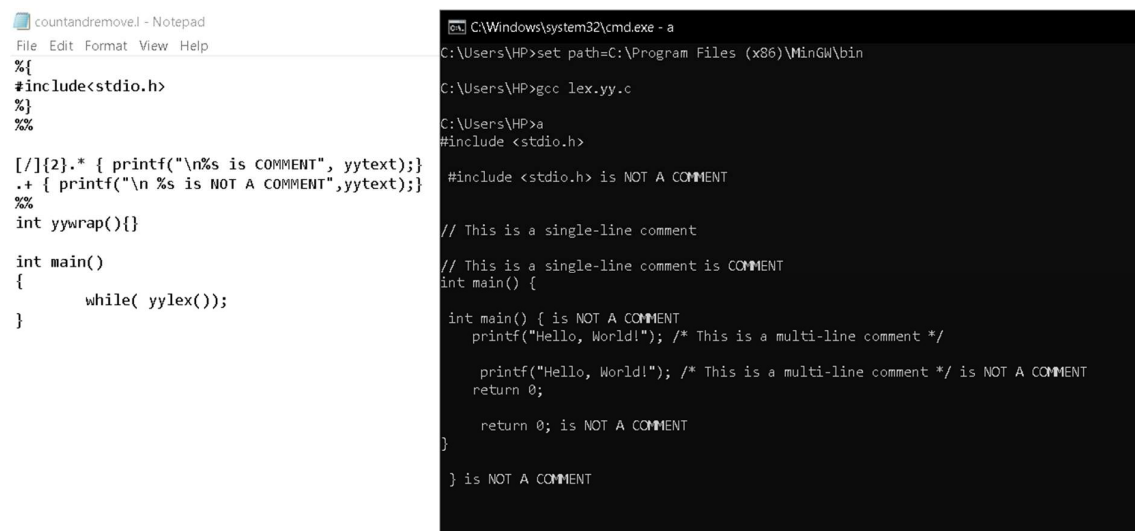
Aim:

To write a LEX specification file that counts the number of comment lines and removes them from a given C program.

Code:

```
%{
#include <stdio.h>
int comment_count = 0;
}%
%{
"//".* { comment_count++; }
"/*"([^\*]|\"*\*/)\"*\*/" { comment_count++; }
.\n { printf("%s", yytext); }
}%
int main() {
    yylex();
    printf("\nNumber of Comments Removed: %d\n", comment_count);
    return 0;
}
```

Output:



The image shows two side-by-side windows. The left window is a Notepad editor titled 'countandremove.l - Notepad'. It contains the LEX specification code shown in the 'Code' section. The right window is a Windows command prompt titled 'C:\Windows\system32\cmd.exe - a'. It shows the execution of the LEX program. The prompt shows the user setting the path to the MinGW bin directory, compiling the LEX program into 'lex.yy.c', and then running it. The output of the program is displayed in the command prompt, showing the number of comments removed and the resulting C code with comments removed.

```
countandremove.l - Notepad
File Edit Format View Help
%{
#include<stdio.h>
}%
%{
[/]{2}.* { printf("\n%s is COMMENT", yytext);}
.+ { printf("\n %s is NOT A COMMENT",yytext);}
}%
int yywrap(){}

int main()
{
    while( yylex());
}
```

```
C:\Windows\system32\cmd.exe - a
C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin
C:\Users\HP>gcc lex.yy.c
C:\Users\HP>a
#include <stdio.h>
#include <stdio.h> is NOT A COMMENT

// This is a single-line comment
// This is a single-line comment is COMMENT
int main() {
int main() { is NOT A COMMENT
    printf("Hello, World!"); /* This is a multi-line comment */
    printf("Hello, World!"); /* This is a multi-line comment */ is NOT A COMMENT
    return 0;
    return 0; is NOT A COMMENT
}
} is NOT A COMMENT
```

12. LEX Program to Convert "abc" to "ABC" in an Input String

Aim:

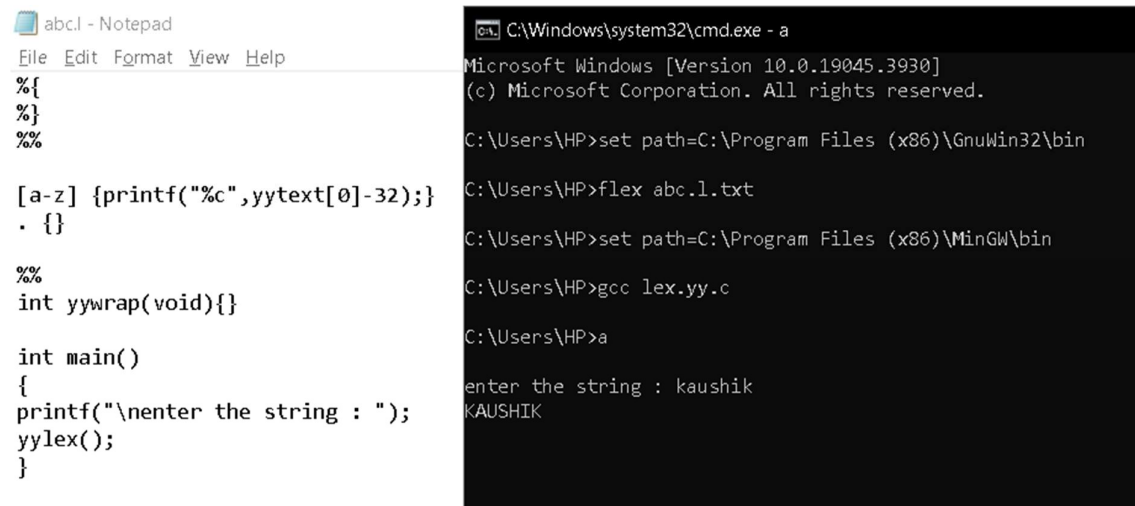
To write a LEX program that replaces every occurrence of the substring "abc" with "ABC" in the given input.

Code:

```
%{
#include <stdio.h>
%}
%%
abc { printf("ABC"); }
.|n { printf("%s", yytext); } // Print other characters as they are
%%

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

Output:



The image shows two side-by-side windows. The left window is a Notepad editor titled 'abc.l - Notepad' with a menu bar (File, Edit, Format, View, Help). It contains the following C code:

```
%{
%}
%%

[a-z] {printf("%c", yytext[0]-32);}
. {}

%%
int yywrap(void){}

int main()
{
printf("\nenter the string : ");
yylex();
}
```

The right window is a Windows Command Prompt titled 'C:\Windows\system32\cmd.exe - a'. It shows the following commands and output:

```
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin

C:\Users\HP>flex abc.l.txt

C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin

C:\Users\HP>gcc lex.yy.c

C:\Users\HP>a

enter the string : kaushik
KAUSHIK
```

13. Implementing a Lexical Analyzer using FLEX

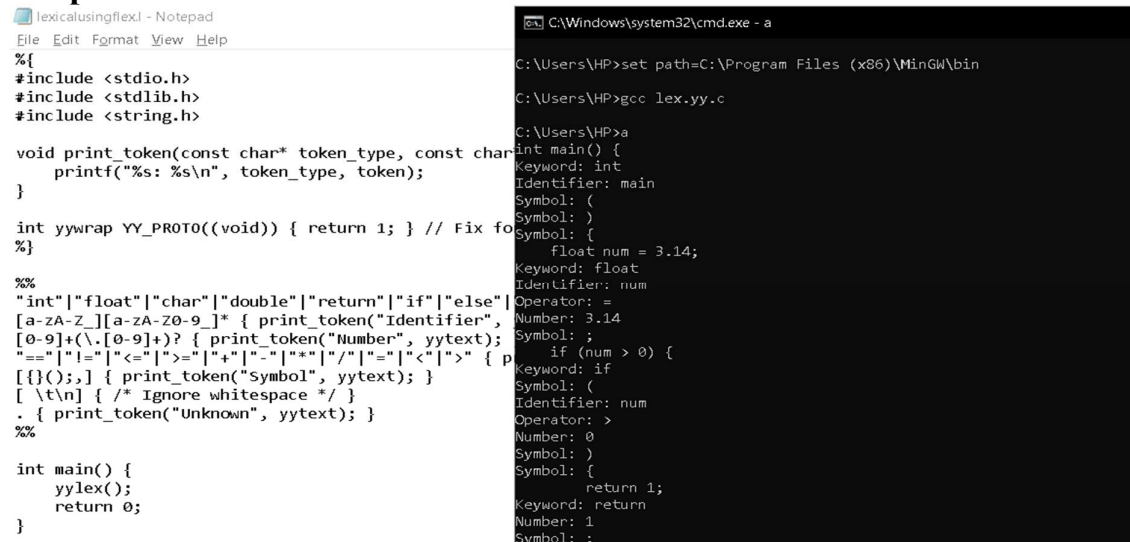
Aim:

To write a FLEX program that tokenizes a given C program by identifying keywords, identifiers, operators, numbers, and symbols.

Code :

```
%{
#include <stdio.h>
#include <string.h>
void print_token(const char* token_type, const char* token) {
    printf("%s: %s\n", token_type, token);
}
}%
%%
"int"|"float"|"char"|"double"|"return"|"if"|"else"|"while"|"for"|"void" {
    print_token("Keyword", yytext); }
[a-zA-Z_][a-zA-Z0-9_]* { print_token("Identifier", yytext); }
[0-9]+(\.[0-9]+)? { print_token("Number", yytext); }
"=="|"!="|"<="|">="|"+"| "-"|"*"|"/"|"="|"<"|">" { print_token("Operator",
yytext); }
[{}();,] { print_token("Symbol", yytext); }
[ \t\n] { /* Ignore whitespace */ }
.
{ print_token("Unknown", yytext); }
}%
int main() {
    yylex();
    return 0;
}
```

Output:



The image shows two windows side-by-side. The left window is a Notepad editor titled 'lexicalusingflex.l - Notepad' containing the FLEX program code. The right window is a Windows Command Prompt titled 'C:\Windows\system32\cmd.exe - a' showing the compilation and execution of the program. The output of the program is displayed in the Command Prompt, showing the tokens identified by the lexical analyzer for the provided C code snippet.

```
C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin
C:\Users\HP>gcc lex.yy.c
C:\Users\HP>a
int main() {
Keyword: int
Identifier: main
Symbol: (
Symbol: )
Symbol: {
    float num = 3.14;
Keyword: float
Identifier: num
Operator: =
Number: 3.14
Symbol: ;
    if (num > 0) {
Keyword: if
Symbol: (
Identifier: num
Operator: >
Number: 0
Symbol: )
Symbol: {
    return 1;
Keyword: return
Number: 1
Symbol: ;
}
```

14. LEX Program to Separate Keywords and Identifiers

Aim:


To write a LEX program that separates keywords and identifiers from an input C program.

Code:

```
%{
#include <stdio.h>
%}
%%
"int"|"float"|"char"|"double"|"return"|"if"|"else"|"while"|"for"|"void" {
printf("Keyword: %s\n", yytext); }
[a-zA-Z_][a-zA-Z0-9_]* { printf("Identifier: %s\n", yytext); }
[\t\n] { /* Ignore whitespace */ }
. { /* Ignore other characters */ }
%%

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

Output:

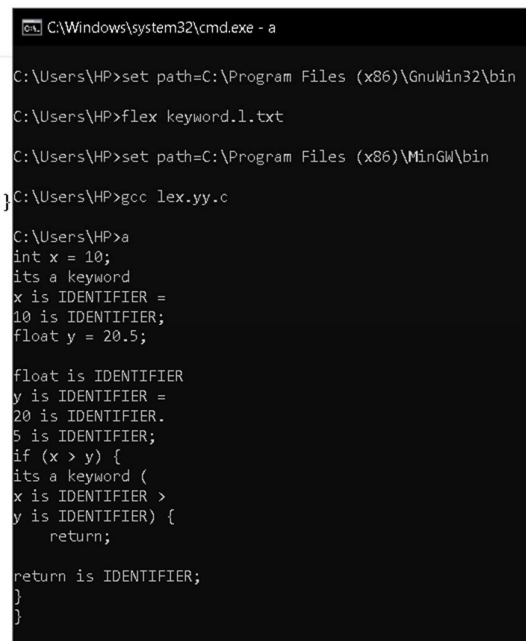


```
keyword.l - Notepad
File Edit Format View Help
%{
#include<stdio.h>
%}

%%

if|else|while|int|switch|for|char { printf("its a keyword");}
[a-zA-Z0-9]+ { printf("\n%s is IDENTIFIER", yytext);}

%%
int yywrap( ){
int main()
{
    while( yylex());
}
```



```
C:\Windows\system32\cmd.exe - a
C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin
C:\Users\HP>flex keyword.l.txt
C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin
C:\Users\HP>gcc lex.yy.c
C:\Users\HP>a
int x = 10;
its a keyword
x is IDENTIFIER =
10 is IDENTIFIER;
float y = 20.5;

float is IDENTIFIER
y is IDENTIFIER =
20 is IDENTIFIER.
5 is IDENTIFIER;
if (x > y) {
its a keyword (
x is IDENTIFIER >
y is IDENTIFIER) {
    return;
}
return is IDENTIFIER;
}
```

15. LEX Program to Recognize Numbers and Words in a Statement

Aim:

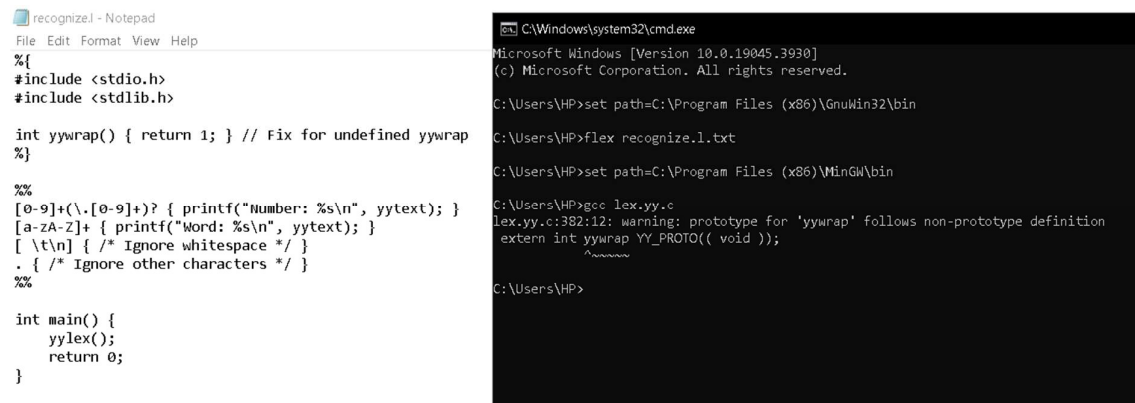
To write a LEX program that identifies numbers and words from a given statement.

Code:

```
%{
#include <stdio.h>
%}
%%
[0-9]+(\.[0-9]+)? { printf("Number: %s\n", yytext); }
[a-zA-Z]+ { printf("Word: %s\n", yytext); }
[ \t\n] { /* Ignore whitespace */ }
. { /* Ignore other characters */ }
%%

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

Output:



The image shows two side-by-side windows. The left window is a Notepad editor titled 'recognize.l - Notepad' containing the LEX program code. The right window is a Windows Command Prompt titled 'C:\Windows\system32\cmd.exe' showing the steps to compile and run the program. The command prompt shows the path being set to 'C:\Program Files (x86)\GnuWin32\bin', the file 'flex recognize.l.txt' being run, the path being set to 'C:\Program Files (x86)\MinGW\bin', and the command 'gcc lex.yy.c' being executed. The output shows a warning about the 'yywrap' prototype and the successful execution of the program.

```
recognize.l - Notepad
File Edit Format View Help
%{
#include <stdio.h>
#include <stdlib.h>

int yywrap() { return 1; } // Fix for undefined yywrap
%}

%%
[0-9]+(\.[0-9]+)? { printf("Number: %s\n", yytext); }
[a-zA-Z]+ { printf("Word: %s\n", yytext); }
[ \t\n] { /* Ignore whitespace */ }
. { /* Ignore other characters */ }
%%

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

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin
C:\Users\HP>flex recognize.l.txt
C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin
C:\Users\HP>gcc lex.yy.c
lex.yy.c:382:12: warning: prototype for 'yywrap' follows non-prototype definition
extern int yywrap YY_PROTO(( void ));
           ^
C:\Users\HP>
```

16. LEX Program to Identify and Count Positive and Negative Numbers

Aim:

To write a LEX program that identifies and counts positive and negative numbers in the input.

Code:

```
%{
#include <stdio.h>
int positive_count = 0, negative_count = 0;
%}
%%
"-[0-9]+ { printf("Negative Number: %s\n", yytext); negative_count++; }
[0-9]+ { printf("Positive Number: %s\n", yytext); positive_count++; }
[\t\n] { /* Ignore whitespace */ }
. { /* Ignore other characters */ }
%%
int main() {
    yylex();
    printf("\nTotal Positive Numbers: %d\nTotal Negative Numbers: %d\n",
    positive_count, negative_count);
    return 0;
}
```

Output:

```
posandneg1 - Notepad
File Edit Format View Help
%{
int positive_no = 0, negative_no = 0;
%}

%%
^-[0-9]+ {negative_no++;
        printf("negative number = %s\n",yytext);}

[0-9]+ {positive_no++;
        printf("positive number = %s\n",yytext);}

%%

int yywrap(){}
int main()
{

    yylex();
    printf ("number of positive numbers = %d,"
            "number of negative numbers = %d\n",
            positive_no, negative_no);

    return 0;
}
```

```
C:\Windows\system32\cmd.exe - a
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin
C:\Users\HP>flex posandneg.1.txt
C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin
C:\Users\HP>gcc lex.yy.c
C:\Users\HP>a
23 -45 67 -89 100
positive number = 23
-positive number = 45
positive number = 67
-positive number = 89
positive number = 100
```


17. LEX Program to Validate a URL

Aim:

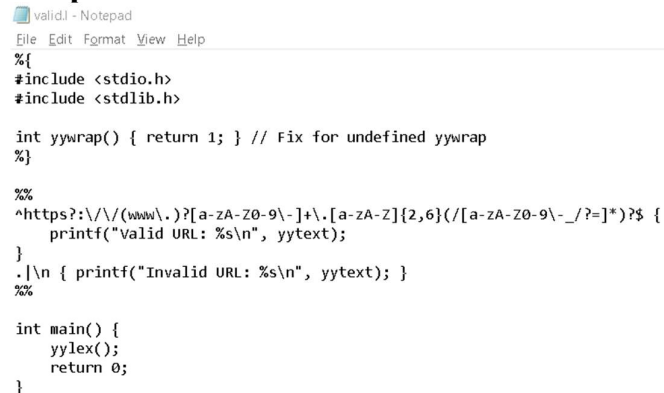
To write a LEX program that validates whether an input string is a valid URL.

Code:

```
%{
#include <stdio.h>
%}
%%
^https?:\\(www\\.)?[a-zA-Z0-9\\-]+\\.[a-zA-Z]{2,6}(/[a-zA-Z0-9\\-_/=?]*)?$ {
    printf("Valid URL: %s\\n", yytext);
}
.|\\n { printf("Invalid URL: %s\\n", yytext); }
%%

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

Output:

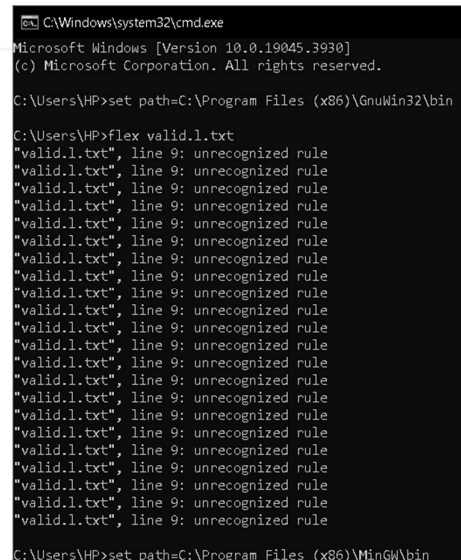


```
valid.l - Notepad
File Edit Format View Help
%{
#include <stdio.h>
#include <stdlib.h>

int yywrap() { return 1; } // Fix for undefined yywrap
%}

%%
^https?:\\(www\\.)?[a-zA-Z0-9\\-]+\\.[a-zA-Z]{2,6}(/[a-zA-Z0-9\\-_/=?]*)?$ {
    printf("Valid URL: %s\\n", yytext);
}
.|\\n { printf("Invalid URL: %s\\n", yytext); }
%%

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



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin

C:\Users\HP>flex valid.l.txt
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
"valid.l.txt", line 9: unrecognized rule
C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin
```

18. LEX Program to Validate Student DOB (Format: DD/MM/YYYY)

Aim:

To write a LEX program that validates the Date of Birth (DOB) format as DD/MM/YYYY.

Code:

```
%{
#include <stdio.h>
%}
%%
[0-3][0-9]/[0-1][0-9]/[0-9]{4} { printf("Valid DOB: %s\n", yytext); }
.|\\n { printf("Invalid DOB: %s\n", yytext); }
%%
int main() {
    yylex();
    return 0;
}
```

Output:

dob.l - Notepad

File Edit Format View Help

```
%{
#include<stdio.h>
%}
```

```
%%
```

```
[0-9][0-9]\\/[0-1][0-9]\\/[1-2][0-9]{3} { printf("valid");}
.+ { printf("invalid");}
```

```
%%
```

```
int yywrap(){}
```

```
int main()
{
    yylex();
}
```

C:\Windows\system32\cmd.exe - a

Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin

C:\Users\HP>flex dob.l.txt

C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin

C:\Users\HP>gcc lex.yy.c

C:\Users\HP>a

31-08-2005

invalid

31/08/2005

valid

19. LEX Program to Implement Basic Mathematical Operations

Aim:

To write a LEX program that performs basic mathematical operations (+, -, *, /).

Code:

```
%{
#include <stdio.h>
int result = 0, num1 = 0, num2 = 0;
char op;
%}
%%
[0-9]+ {
    if (num1 == 0)
        num1 = atoi(yytext);
    else
        num2 = atoi(yytext);
}
[+\-*/] { op = yytext[0]; }
\n {
    switch(op) {
        case '+': result = num1 + num2; break;
        case '-': result = num1 - num2; break;
        case '*': result = num1 * num2; break;
        case '/': result = (num2 != 0) ? num1 / num2 : 0; break;
        default: result = 0;
    }
    printf("Result: %d %c %d = %d\n", num1, op, num2, result);
    num1 = num2 = 0; // Reset
}
%%
int main() {
    yylex();
    return 0;
}
```

Output:

math.l - Notepad

File Edit Format View Help

```
%{
#include<stdio.h>
%}
%%

"="|"+"| "-"| "/"| "*" { printf("valid");}
.+ {printf("invalid");}
%%

int yywrap(){}
int main()
{
printf("enter the input:");
yylex();
return 0;
}
```

Select C:\Windows\system32\cmd.exe

C:\Users\HP>set path=C:\Program Files (x86)\GnuWin32\bin

C:\Users\HP>flex math.l.txt

C:\Users\HP>set path=C:\Program Files (x86)\MinGW\bin

C:\Users\HP>gcc lex.yy.c

C:\Users\HP>a

enter the input:12 + 5

invalid

8 * 3

invalid

20 - 7

invalid

15 / 3

invalid