INDEX

Experiment No: 01

Name of the Experiment

Write a simple lex specification to recognize the following verb.

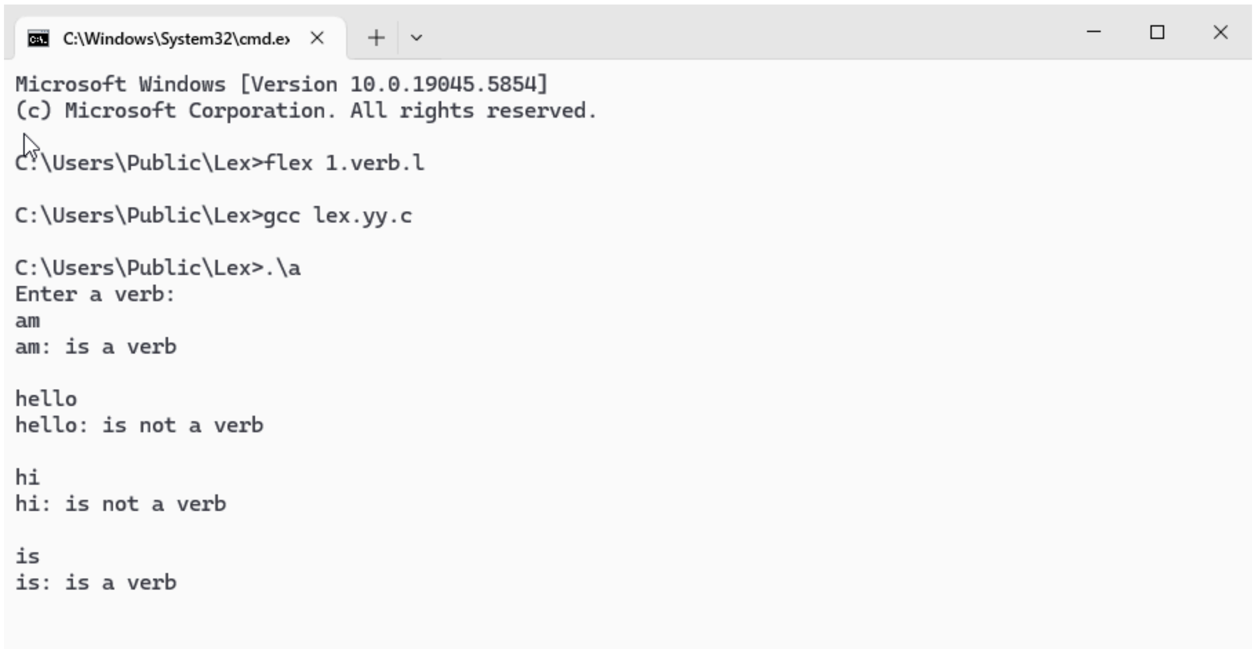
Objective:

To develop a Lex program that identifies and recognizes verbs from a given input text using lexical analysis.

Code:

```
%option noyywrap
%{
 /*Define section*/
  #include <stdio.h>
%}


/*Rule section*/
%%
[\t ]+ ( is |  am | are | were | was | be | has | have | had | go)
{ printf("%s: is a verb\n",yytext); }

[a-zA-Z]+ {printf("%s: is not a verb\n",yytext);}
.|\n {ECHO;/*normal default anyway*/}
%%

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

Output of Experiment 01:



```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Public\Lex>flex 1.verb.l

C:\Users\Public\Lex>gcc lex.yy.c

C:\Users\Public\Lex>.\a
Enter a verb:
am
am: is a verb

hello
hello: is not a verb

hi
hi: is not a verb

is
is: is a verb
```

Experiment No: 02

Name of the Experiment

Write a simple lex specification to recognize the following words as different parts of speech

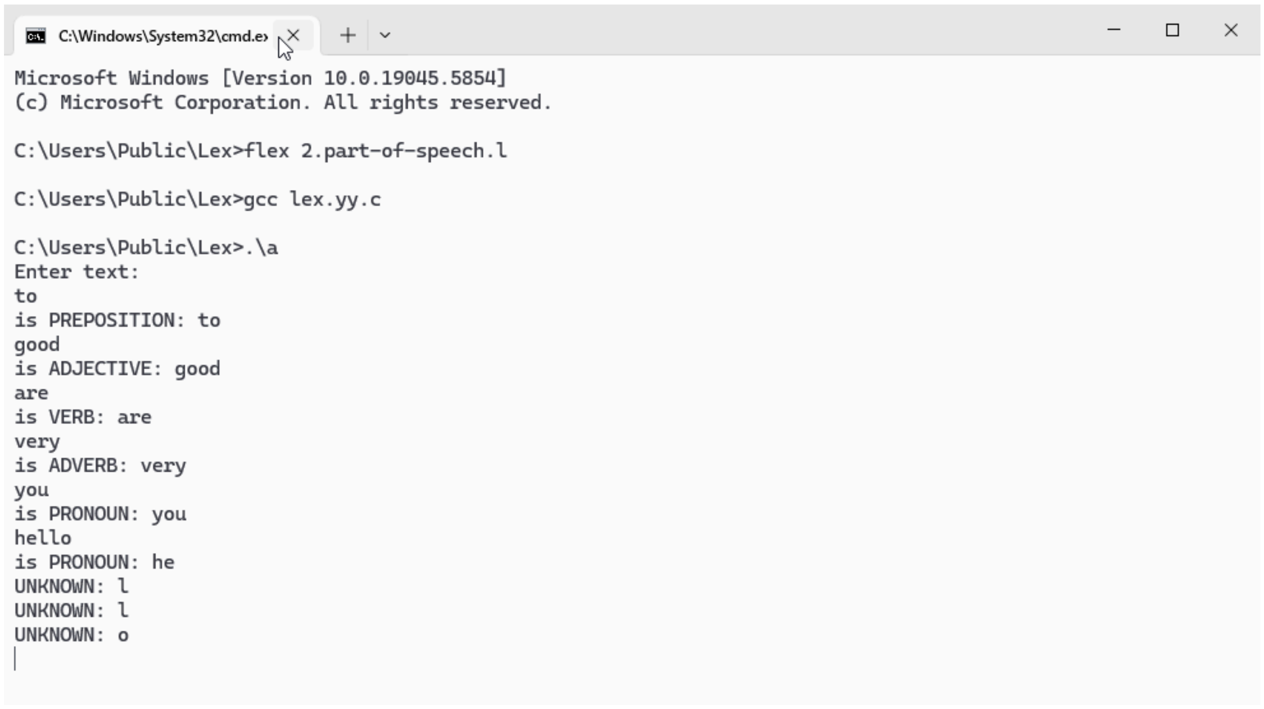
Objective:

To write a simple Lex program that identifies and classifies given words into different parts of speech by using lexical analysis techniques.

Code:

```
%option noyywrap

%{
  #include <stdio.h>
%}

%%
is|am|are|were|go              { printf("VERB: %s\n", yytext); }
very|simply|quickly|gently     { printf("ADVERB: %s\n", yytext); }
to|from|behind|between         { printf("PREPOSITION: %s\n", yytext); }
if|then                        { printf("CONJUNCTION: %s\n", yytext); }
Good|bad|small|beautiful|happy { printf("ADJECTIVE: %s\n", yytext); }
I|you|he|she|we|it|me|us|him|her { printf("PRONOUN: %s\n", yytext); }
[ \t\n]+
. { printf("UNKNOWN: %s\n", yytext); }
%%

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

Output of Experiment 02:



C:\Windows\System32\cmd.e

```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Public\Lex>flex 2.part-of-speech.l

C:\Users\Public\Lex>gcc lex.yy.c

C:\Users\Public\Lex>.\a
Enter text:
to
is PREPOSITION: to
good
is ADJECTIVE: good
are
is VERB: are
very
is ADVERB: very
you
is PRONOUN: you
hello
is PRONOUN: he
UNKNOWN: l
UNKNOWN: l
UNKNOWN: o
```

Experiment No: 03

Name of the Experiment

Write a simple lex specification to recognize different keywords.

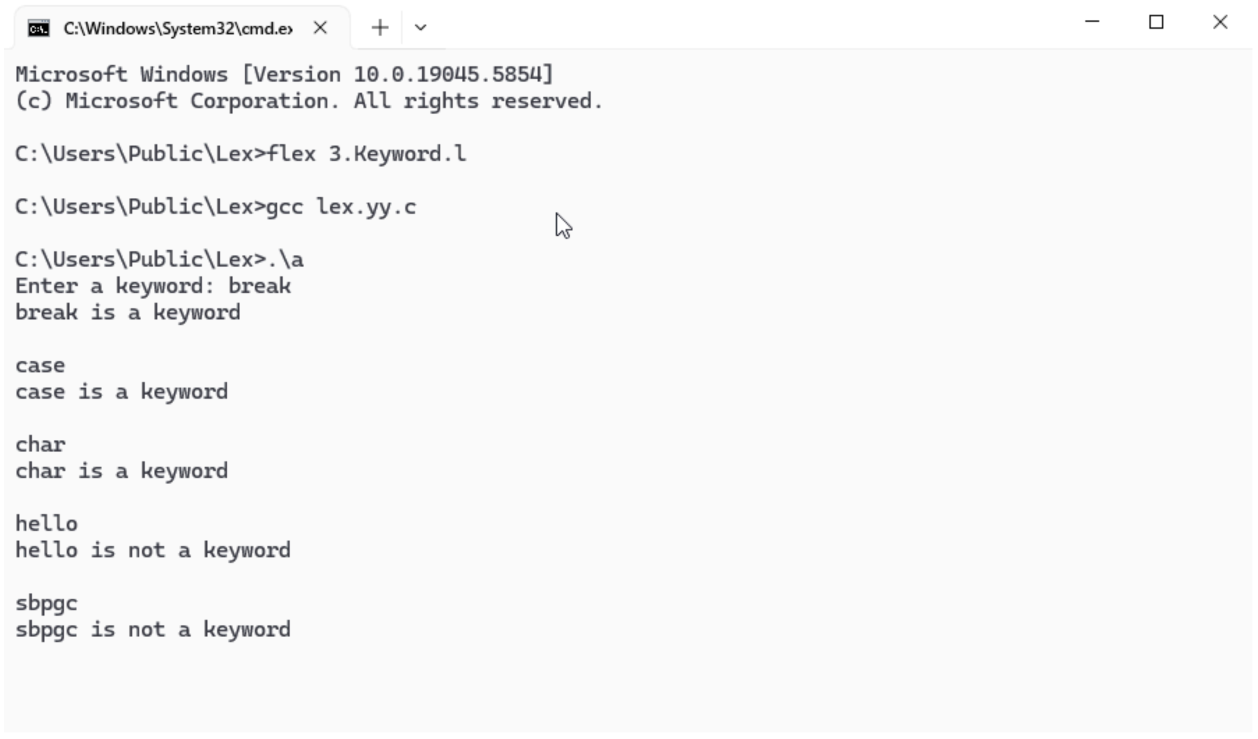
Objective:

To develop a Lex program that recognizes and classifies different keywords based on pattern matching during lexical analysis.

Code:

```
%option noyywrap

%{
  #include <stdio.h>
%}

%%
[\t]+
(auto | break |case | char | const | continue | default | do | double | else | enum | extern | float | for|goto
| if|int | long|register | return | short | signed | sizeof | static|struct | switch | typedef | union|unsigned |
void | volatile | while | go ) { printf("%s is a keyword\n",yytext); }

[a-zA-Z]+ {printf("%s is not a keyword\n",yytext);}
[0-9]+ {printf("%s is a number\n",yytext);}
.|\n {ECHO;/* Normal default action */}

%%
int main()
{
  printf("Enter a keyword: ");
  yylex();
  return 0;
}
```

Output of Experiment 03:



```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Public\Lex>flex 3.Keyword.l

C:\Users\Public\Lex>gcc lex.yy.c

C:\Users\Public\Lex>.\a
Enter a keyword: break
break is a keyword

case
case is a keyword

char
char is a keyword

hello
hello is not a keyword

sbpgc
sbpgc is not a keyword
```

Experiment No:04

Name of the Experiment

Write a simple lex specification to recognize the identifier.

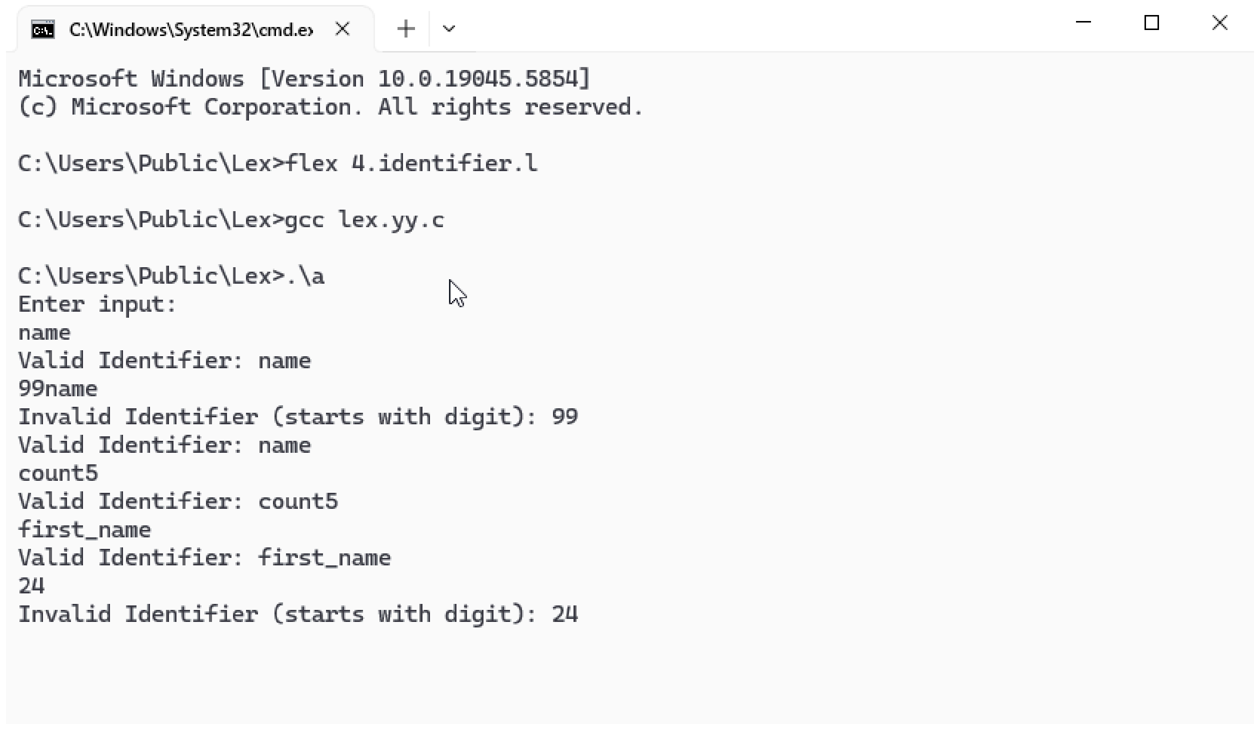
Objective:

To write a simple Lex program that recognizes identifiers in a source code using lexical analysis and pattern matching.

Code:

```
%option noyywrap

%{
#include <stdio.h>
%}

/* Rule Section */
%%
[a-zA-Z_][a-zA-Z0-9_]*     { printf("Valid Identifier: %s\n", yytext); }
[0-9]+                     { printf("Invalid Identifier: %s\n", yytext); }
[^a-zA-Z0-9_\n\t ]+        { printf("Invalid Symbol: %s\n", yytext); }
[ \t\n]+                   ;

%%
int main(void) {
    printf("Enter input:\n");
    yylex();
    return 0;
}
```

Output of Experiment 04:



```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Public\Lex>flex 4.identifier.l

C:\Users\Public\Lex>gcc lex.yy.c

C:\Users\Public\Lex>.\a
Enter input:
name
Valid Identifier: name
99name
Invalid Identifier (starts with digit): 99
Valid Identifier: name
count5
Valid Identifier: count5
first_name
Valid Identifier: first_name
24
Invalid Identifier (starts with digit): 24
```

Experiment No:05

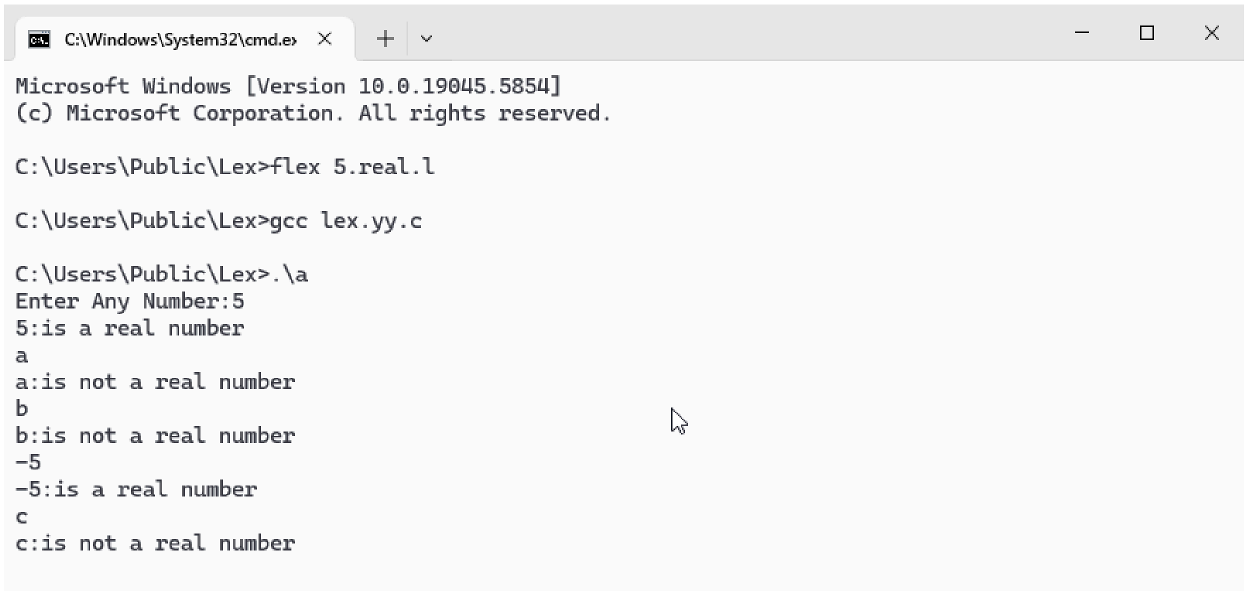
Name of the Experiment

Write a simple lex specification to recognize real numbers.

Code:

```
%option noyywrap

%{
 #include <stdio.h>
%}



%%
[-]*[0-9]+|[0-9]\.[0-9] {printf("%s:is a real number",yytext);}
.* {printf("%s:is not a real number",yytext);}
%%


int main()
{
 printf("Enter Any Number:");
 yylex();
 return 0;

}
```

Output of Experiment 05:


C:\Windows\System32\cmd.ex

```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Public\Lex>flex 5.real.l

C:\Users\Public\Lex>gcc lex.yy.c

C:\Users\Public\Lex>.\a
Enter Any Number:5
5:is a real number
a
a:is not a real number
b
b:is not a real number
-5
-5:is a real number
c
c:is not a real number
```

Experiment No:06

Name of the Experiment

Write a simple lex specification to recognize an integer.

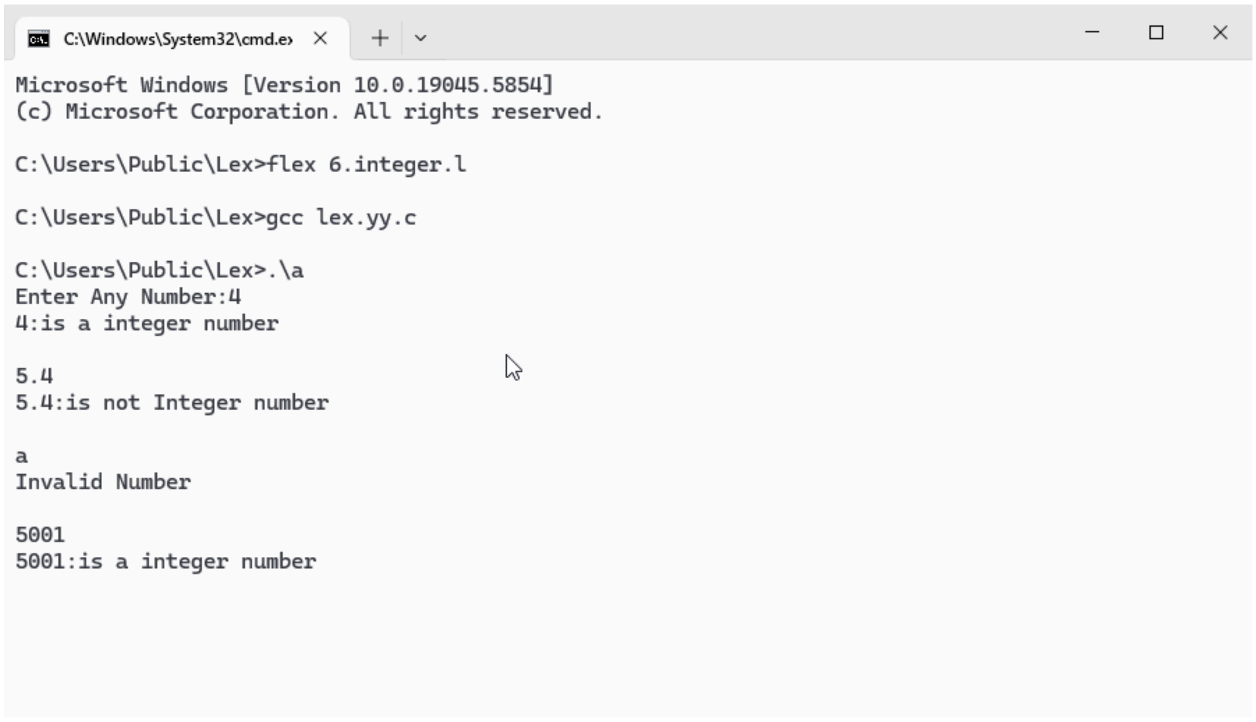
Objective:

To develop a Lex program that recognizes integer numbers from the input using pattern matching in lexical analysis.

Code:

```
%option noyywrap

%{
  #include<stdio.h>
%}

%%
[0-9]+            {printf("%s:is a integer number\n",yytext);}
[0-9]*[.][0-9]+ {printf("%s:is not Integer number\n",yytext);}
.* {printf("Invalid Number\n");}
%%

int main()
{
 printf("Enter Any Number:");
 yylex();
 return 0;
}
```

Output of Experiment 06:



```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Public\Lex>flex 6.integer.l

C:\Users\Public\Lex>gcc lex.yy.c

C:\Users\Public\Lex>.\a
Enter Any Number:4
4:is a integer number

5.4
5.4:is not Integer number

a
Invalid Number

5001
5001:is a integer number
```

Experiment No:07

Name of the Experiment

Write a simple lex specification to recognize a floating-point number.

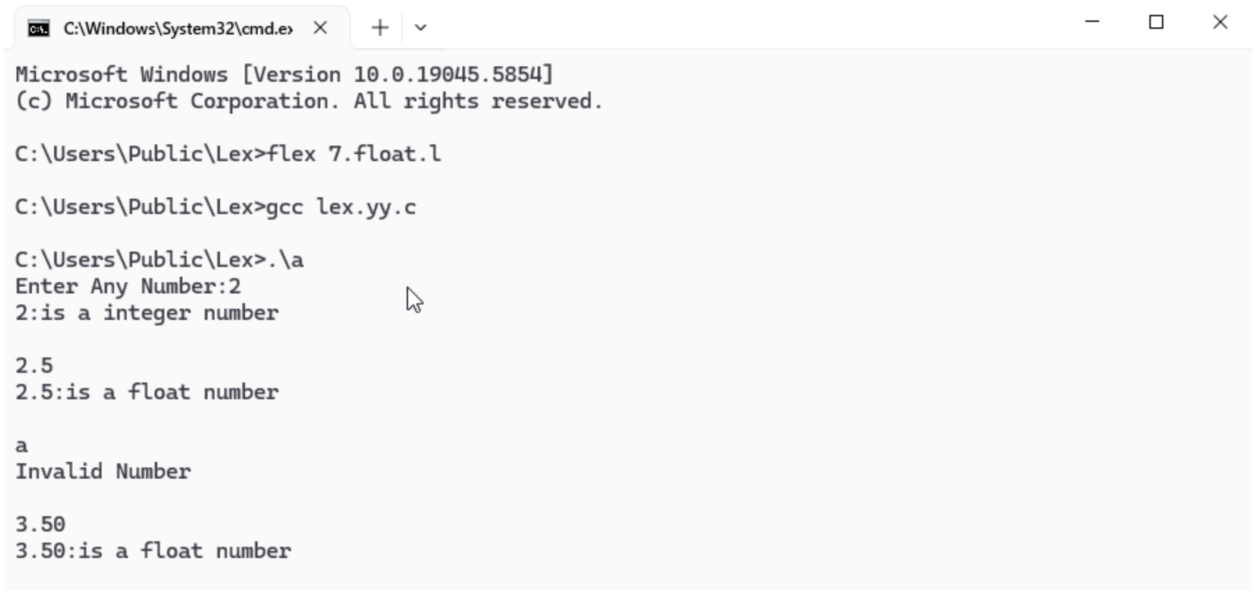
Objective:

Write a Lex program that recognizes floating-point numbers (floats) in the input using pattern matching during lexical analysis.

Code:

```
%option noyywrap

%{
#include<stdio.h>
%}

%%
[0-9]*[.][0-9]+ {printf("%s:is a float number\n",yytext);}
[0-9]*          {printf("%s:is a integer number\n",yytext);}
.* {printf("Invalid Number\n");}
%%

int main()
{
 printf("Enter Any Number:");
 yylex();
 return 0;

}
```

Output of Experiment 07:



C:\Windows\System32\cmd.e›

```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Public\Lex>flex 7.float.l

C:\Users\Public\Lex>gcc lex.yy.c

C:\Users\Public\Lex>.\a
Enter Any Number:2
2:is a integer number

2.5
2.5:is a float number

a
Invalid Number

3.50
3.50:is a float number
```

Experiment No:08

Name of the Experiment

Write a simple lex specification to recognize positive and negative integers and floating numbers.

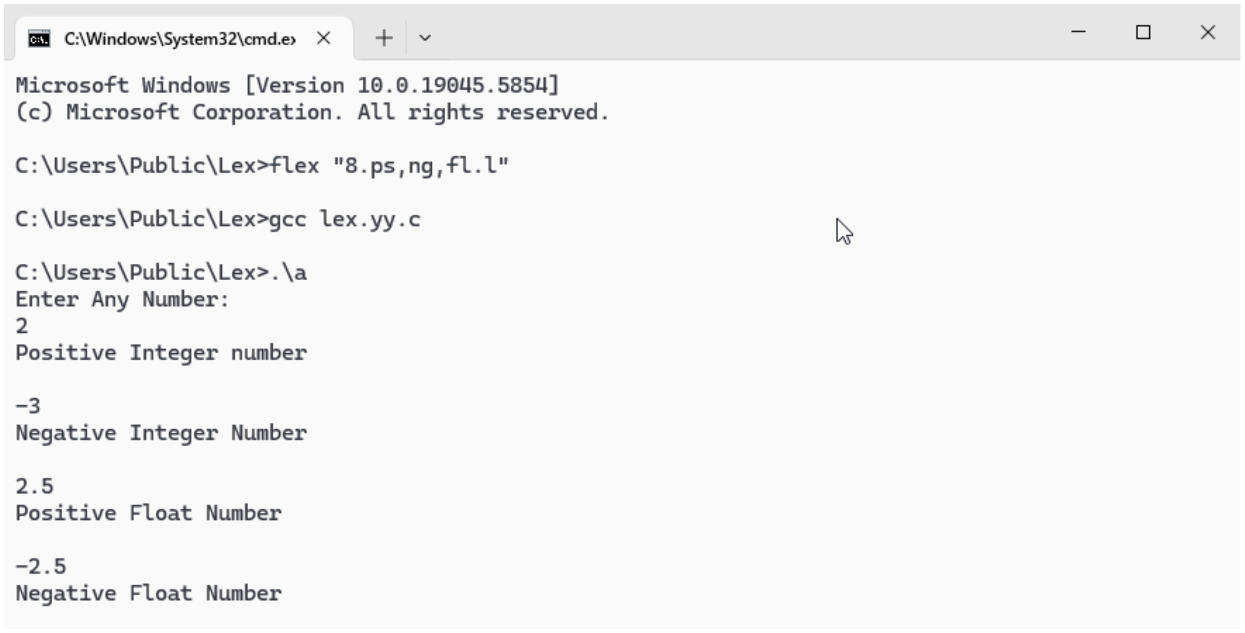
Objective:

To create a Lex program that recognizes positive and negative integers as well as floating-point numbers using pattern matching in lexical analysis.

Code:

```
%option noyywrap

%{
#include<stdio.h>
%}

%%
^-[0-9]+        {printf("Negative Integer Number\n",yytext);}
[+]?[0-9]+      {printf("Positive Integer number\n",yytext);}
^-[0-9]*[.][0-9]+ {printf("Negative Float Number\n",yytext);}
[+]?[0-9]*[.][0-9]+ {printf("Positive Float Number\n",yytext);}
.* {printf("Invalid Number\n");}
%%

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

Output of Experiment 08:


C:\Windows\System32\cmd.ex

```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Public\Lex>flex "8.ps,ng,fl.l"

C:\Users\Public\Lex>gcc lex.yy.c

C:\Users\Public\Lex>.\a
Enter Any Number:
2
Positive Integer number

-3
Negative Integer Number

2.5
Positive Float Number

-2.5
Negative Float Number
```

Experiment No:09

Name of the Experiment

Write a simple lex specification to recognize different punctuation symbols.

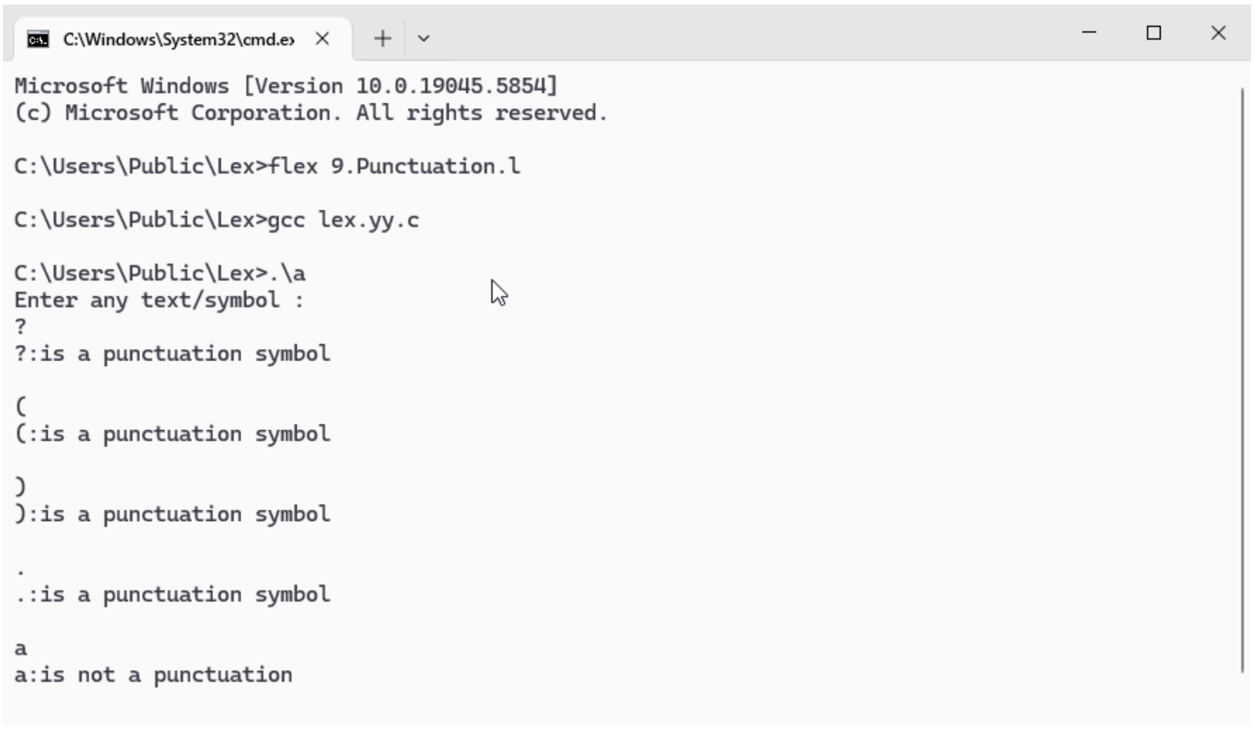
Objective:

To write a Lex program that recognizes and identifies different punctuation symbols using pattern matching during lexical analysis.

Code:

```
%option noyywrap

%{
#include <stdio.h>
%}

%%
[.,;:!?'"(){}\[\]<>\-_@#$%^&*+/=|~`\\] {printf("%s:is a punctuation 
symbol\n",yytext);}
[a-zA-Z0-9] {printf("%s:is not a punctuation\n",yytext);}
%%

int main()
{
 printf("Enter any text/symbol :\n");
 yylex();
 return 0;
}
```

Output of Experiment 09:



```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Public\Lex>flex 9.Punctuation.l

C:\Users\Public\Lex>gcc lex.yy.c

C:\Users\Public\Lex>.\a
Enter any text/symbol :
?
?:is a punctuation symbol

(
(:is a punctuation symbol

)
):is a punctuation symbol

.
.:is a punctuation symbol

a
a:is not a punctuation
```

Experiment No:10

Name of the Experiment

Write a simple lex specification to recognize a digit.

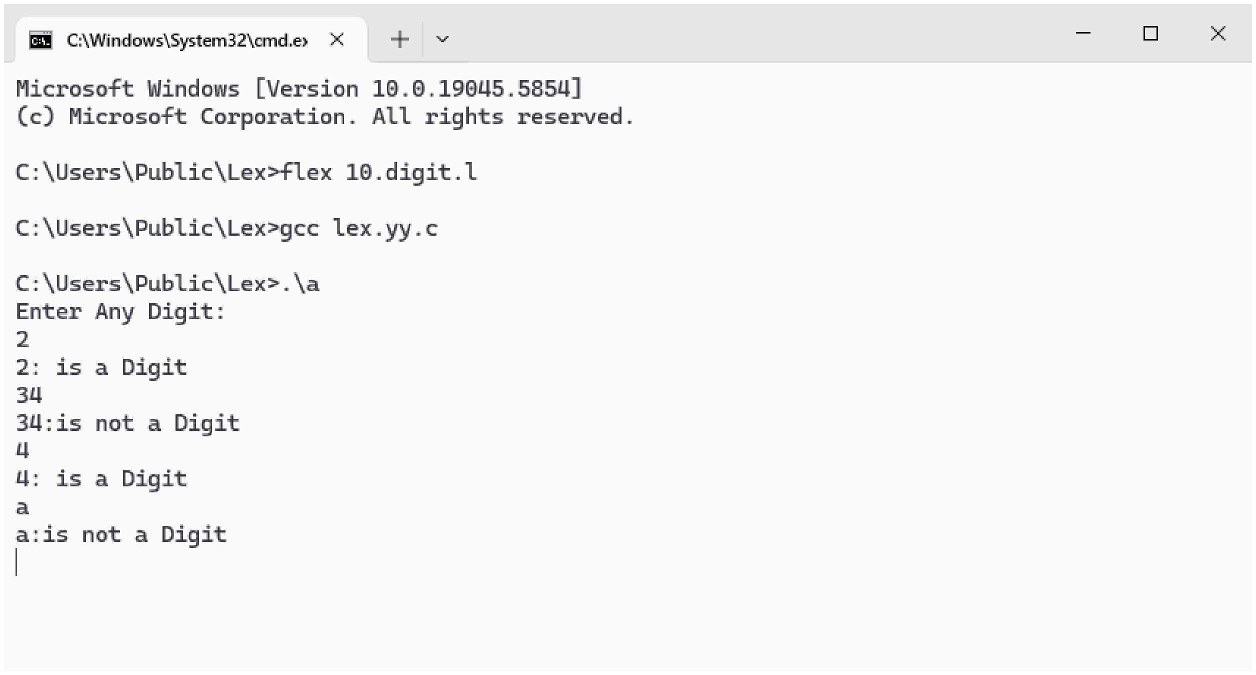
Objective:

To develop a Lex program that recognizes digits (0-9) from the input using pattern matching in lexical analysis.

Code:

```
%option noyywrap

%{
#include<stdio.h>
%}

%%
[0-9] {printf("%s: is a Digit",yytext);}
.* {printf("%s:is not a Digit",yytext);}
%%

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

Output of Experiment 10:



C:\Windows\System32\cmd.ex

```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Public\Lex>flex 10.digit.l

C:\Users\Public\Lex>gcc lex.yy.c

C:\Users\Public\Lex>.\a
Enter Any Digit:
2
2: is a Digit
34
34:is not a Digit
4
4: is a Digit
a
a:is not a Digit
```