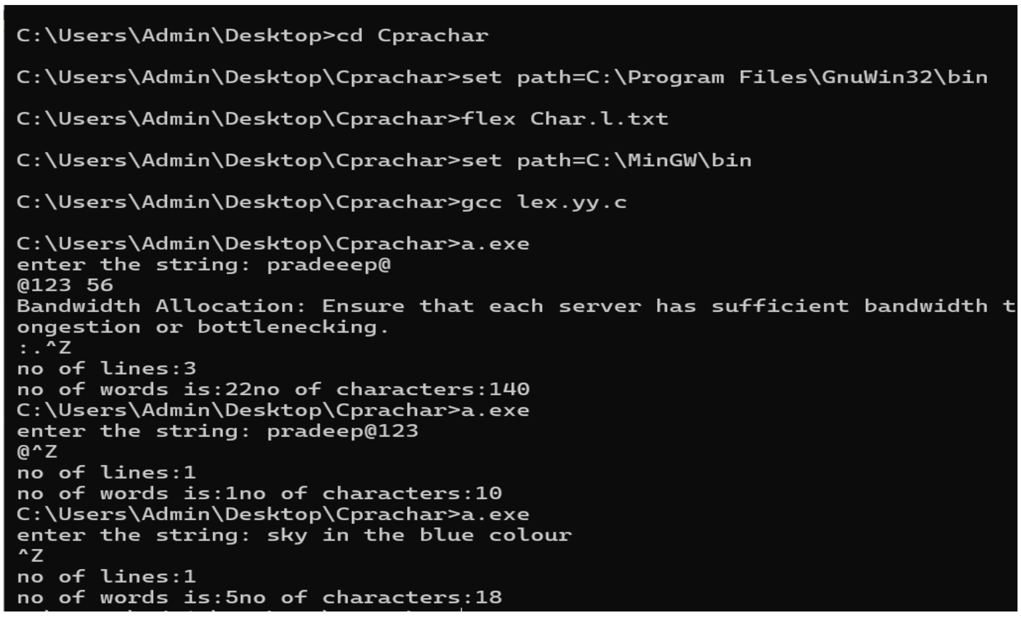
DAY-3 EXPERIMENTS

17. LEX PROGRAM FOR NO.OF CHARS,LINES,WORDS

Program:

```
%{
int i =0,l=0,c=0;
%}
%%
[\n] {l++;}
[ ] {i++;}
[a-zA-Z0-9] {c++;}
%%
int yywrap(){}
int main()
{
printf("enter the string: ");
yylex();
printf("no of lines:%d\n",l);
printf("no of words is:%d",i+l);
printf("no of characters:%d",c);
}
```

Output:



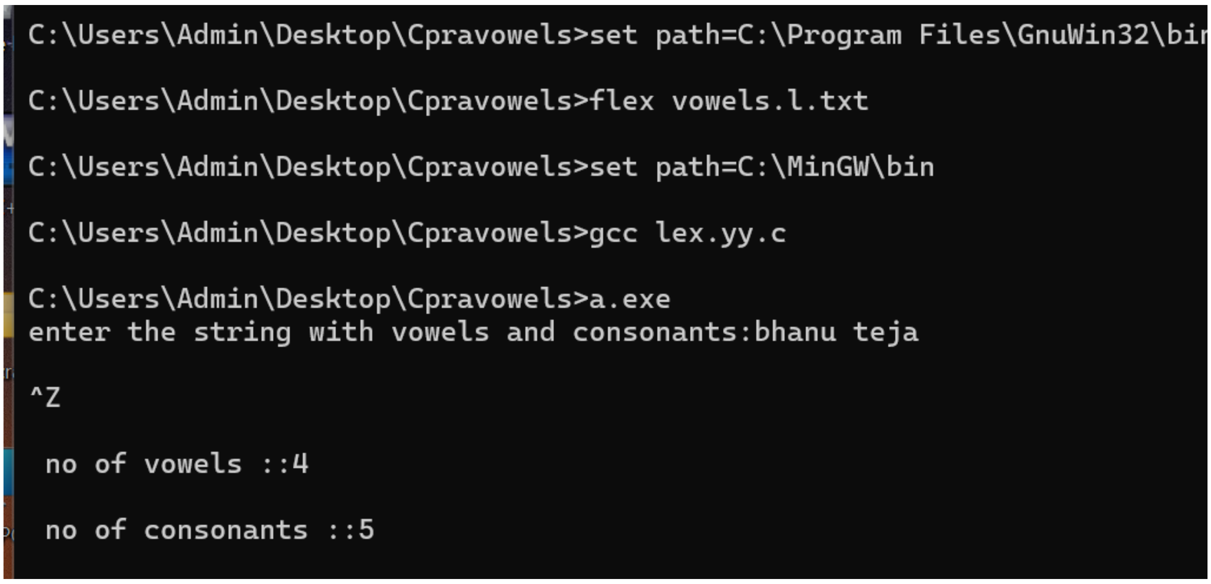
```
C:\Users\Admin\Desktop>cd Cprachar

C:\Users\Admin\Desktop\Cprachar>set path=C:\Program Files\GnuWin32\bin

C:\Users\Admin\Desktop\Cprachar>flex Char.l.txt

C:\Users\Admin\Desktop\Cprachar>set path=C:\MinGW\bin

C:\Users\Admin\Desktop\Cprachar>gcc lex.yy.c

C:\Users\Admin\Desktop\Cprachar>a.exe
enter the string: pradeeep@
@123 56
Bandwidth Allocation: Ensure that each server has sufficient bandwidth t
ongestion or bottlenecking.
:.^Z
no of lines:3
no of words is:22no of characters:140
C:\Users\Admin\Desktop\Cprachar>a.exe
enter the string: pradeep@123
@^Z
no of lines:1
no of words is:1no of characters:10
C:\Users\Admin\Desktop\Cprachar>a.exe
enter the string: sky in the blue colour
^Z
no of lines:1
no of words is:5no of characters:18
```

18. LEX PROGRAM FOR COUNT VOWELS AND CONSONANTS

Program:

```
%{
int vcount=0;
int ccount=0;
%}
%%
[aeiouAEIOU] {vcount++;}
[a-z,A-Z] {ccount++;}
%%
int yywrap(){}
int main()
{
printf("enter the string with vowels and consonants:");
yylex();
printf("\n no of vowels ::%d \n",vcount);
printf("\n no of consonants ::%d \n",ccount);
}
```

Output:



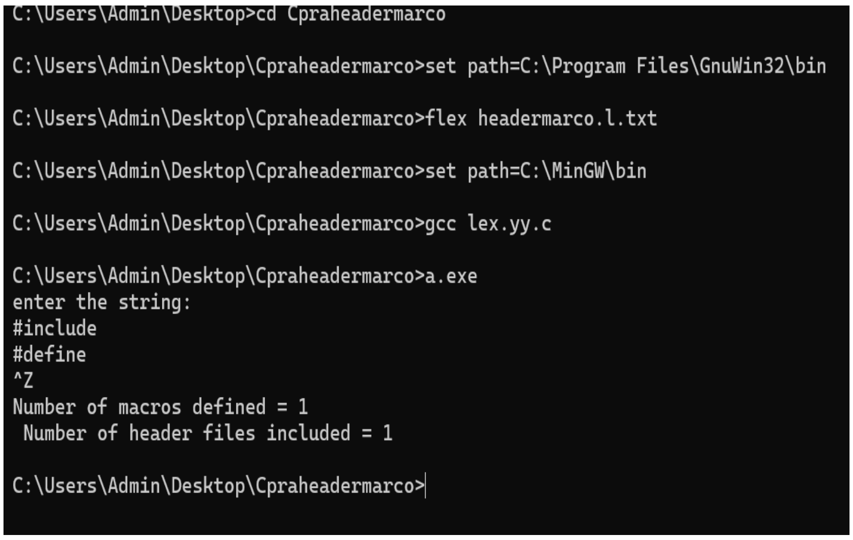
```
C:\Users\Admin\Desktop\Cpravowels>set path=C:\Program Files\GnuWin32\bi

C:\Users\Admin\Desktop\Cpravowels>flex vowels.l.txt

C:\Users\Admin\Desktop\Cpravowels>set path=C:\MinGW\bin

C:\Users\Admin\Desktop\Cpravowels>gcc lex.yy.c

C:\Users\Admin\Desktop\Cpravowels>a.exe
enter the string with vowels and consonants:bhanu teja

^Z

 no of vowels ::4

 no of consonants ::5
```

19.Write a LEX program to count the number of Macros defined and header files

Program:

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

Output:



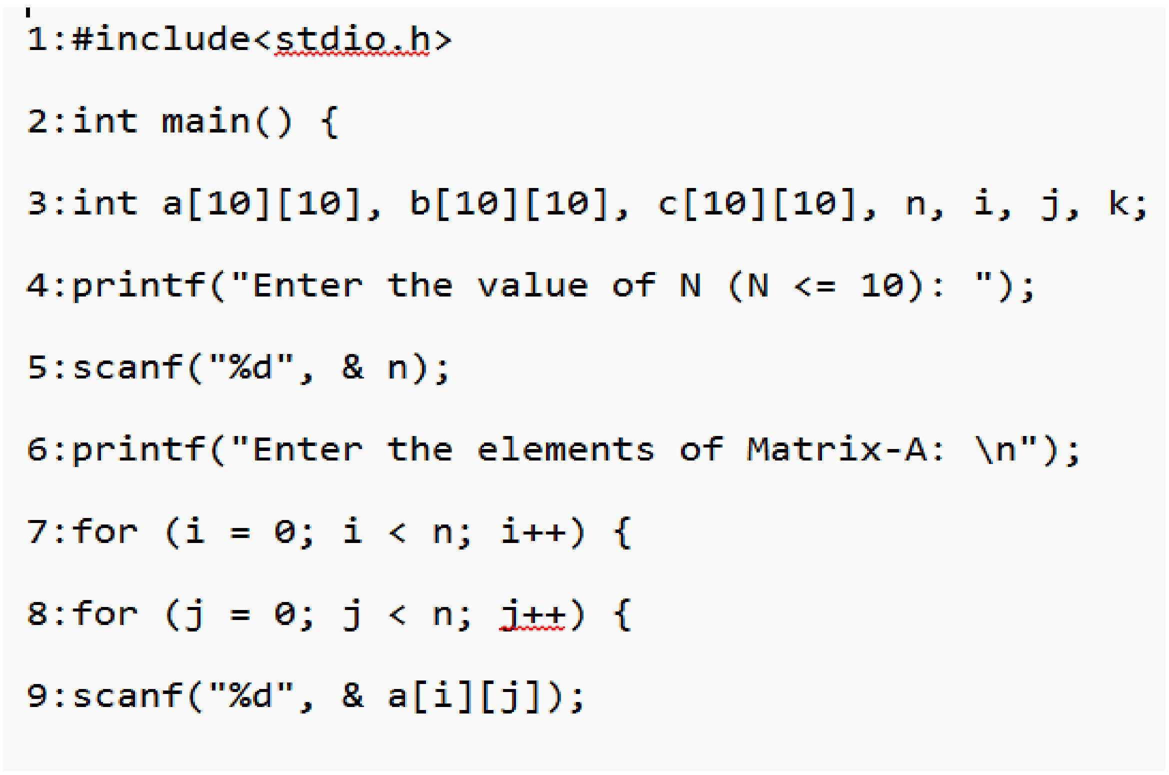
```
C:\Users\Admin\Desktop>cd Cpraheadermarco

C:\Users\Admin\Desktop\Cpraheadermarco>set path=C:\Program Files\GnuWin32\bin

C:\Users\Admin\Desktop\Cpraheadermarco>flex headermarco.l.txt

C:\Users\Admin\Desktop\Cpraheadermarco>set path=C:\MinGW\bin

C:\Users\Admin\Desktop\Cpraheadermarco>gcc lex.yy.c

C:\Users\Admin\Desktop\Cpraheadermarco>a.exe
enter the string:
#include
#define
^Z
Number of macros defined = 1
 Number of header files included = 1

C:\Users\Admin\Desktop\Cpraheadermarco>
```

20. Write a LEX program which adds line numbers to the given C program file and display the same in the standard

Program:

```
%{
int ln=0;
%}
%%
.* {ln++; fprintf(yyout,"\n%d:%s",ln,yytext);}
%%
int yywrap(){}
int main()
{
yyin=fopen("simple1.txt","r");
yyout=fopen("out.txt","w");
yylex();
}
```

Output:



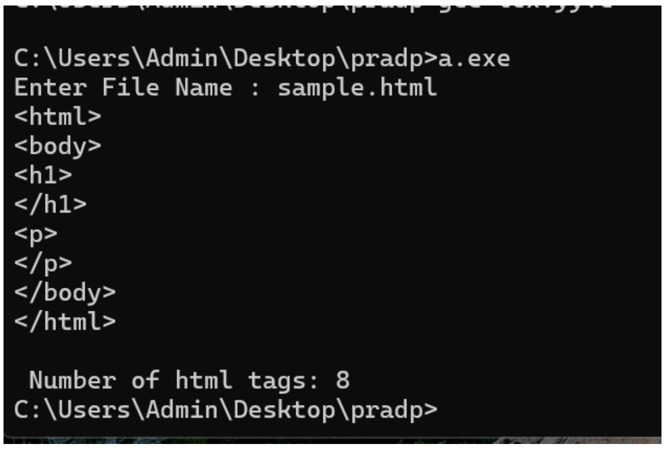
```
1:#include<stdio.h>

2:int main() {

3:int a[10][10], b[10][10], c[10][10], n, i, j, k;

4:printf("Enter the value of N (N <= 10): ");

5:scanf("%d", & n);

6:printf("Enter the elements of Matrix-A: \n");

7:for (i = 0; i < n; i++) {

8:for (j = 0; j < n; j++) {

9:scanf("%d", & a[i][j]);
```

21. Write a LEX program to print all HTML tags in the input file

Program:

```
 %{int tags;
%}
%%
"<"[^>]*>  { tags++; printf("%s \n", yytext); }
.|\n { }
%%
int yywrap(void) {
return 1; }
int main(void)
{
FILE *f;
char file[10];
printf("Enter File Name : ");
scanf("%s",file);
f = fopen(file,"r");
yyin = f;
yylex();
printf("\n Number of html tags: %d",tags);
fclose(yyin);
}
```

Output:



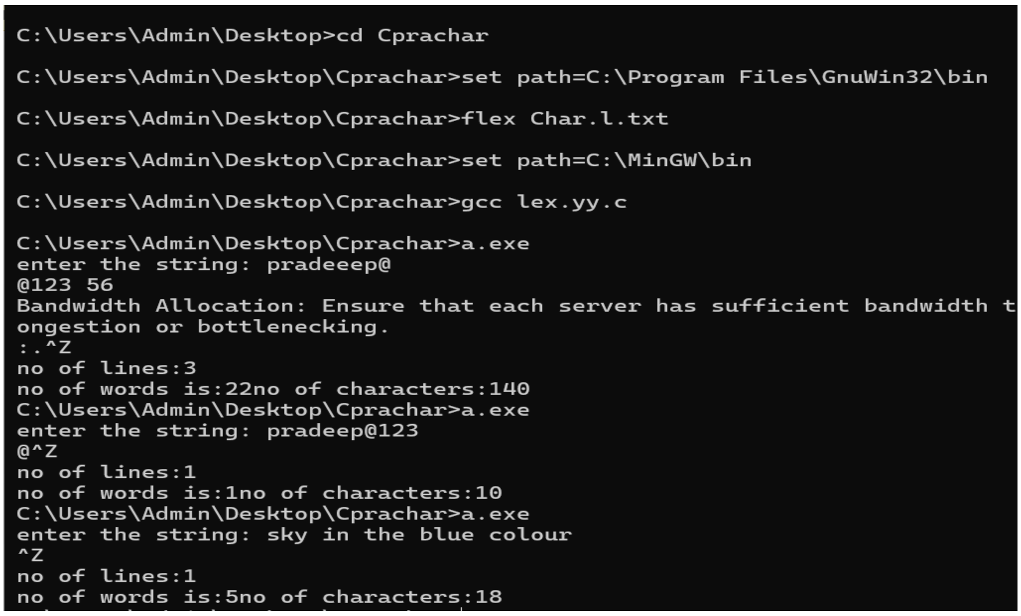
```
C:\Users\Admin\Desktop\pradp>a.exe
Enter File Name : sample.html
<html>
<body>
<h1>
</h1>
<p>
</p>
</body>
</html>

 Number of html tags: 8
C:\Users\Admin\Desktop\pradp>
```

22. Write a LEX specification count the number of characters, number of lines & number of words.

Program:

```
%{
int i =0,l=0,c=0;
%}
%%
[\n] {l++;}
[ ] {i++;}
[a-zA-Z0-9] {c++;}
%%
int yywrap(){}
int main()
{
printf("enter the string: ");
yylex();
printf("no of lines:%d\n",l);
printf("no of words is:%d",i+l);
printf("no of characters:%d",c);
}
```

Output:



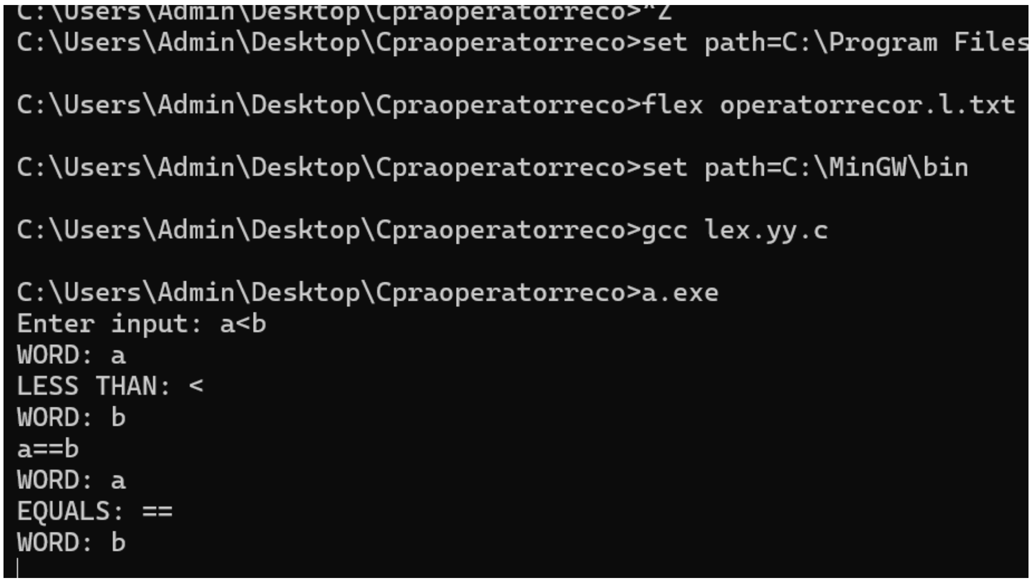
```
C:\Users\Admin\Desktop>cd Cprachar

C:\Users\Admin\Desktop\Cprachar>set path=C:\Program Files\GnuWin32\bin

C:\Users\Admin\Desktop\Cprachar>flex Char.l.txt

C:\Users\Admin\Desktop\Cprachar>set path=C:\MinGW\bin

C:\Users\Admin\Desktop\Cprachar>gcc lex.yy.c

C:\Users\Admin\Desktop\Cprachar>a.exe
enter the string: pradeeep@
@123 56
Bandwidth Allocation: Ensure that each server has sufficient bandwidth t
ongestion or bottlenecking.
:.^Z
no of lines:3
no of words is:22no of characters:140
C:\Users\Admin\Desktop\Cprachar>a.exe
enter the string: pradeep@123
@^Z
no of lines:1
no of words is:1no of characters:10
C:\Users\Admin\Desktop\Cprachar>a.exe
enter the string: sky in the blue colour
^Z
no of lines:1
no of words is:5no of characters:18
```

23. Write LEX program to recognize a word and relational operator.

Program:

```
%{
#include <stdio.h>
%}
%%
[a-zA-Z]+    { printf("WORD: %s\n", yytext); }
"=="         { printf("EQUALS: %s\n", yytext); }
"!="         { printf("NOT EQUALS: %s\n", yytext); }
"<="         { printf("LESS THAN OR EQUAL: %s\n", yytext); }
">="         { printf("GREATER THAN OR EQUAL: %s\n", yytext); }
"<"          { printf("LESS THAN: %s\n", yytext); }
">"          { printf("GREATER THAN: %s\n", yytext); }
[ \t\n]+     { /* Ignore whitespace */ }
.            { printf("UNKNOWN: %s\n", yytext); }
%%

int main() {
    while (1) {
        printf("Enter input: ");
        yylex();
    }
    return 0;
}

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

Output:



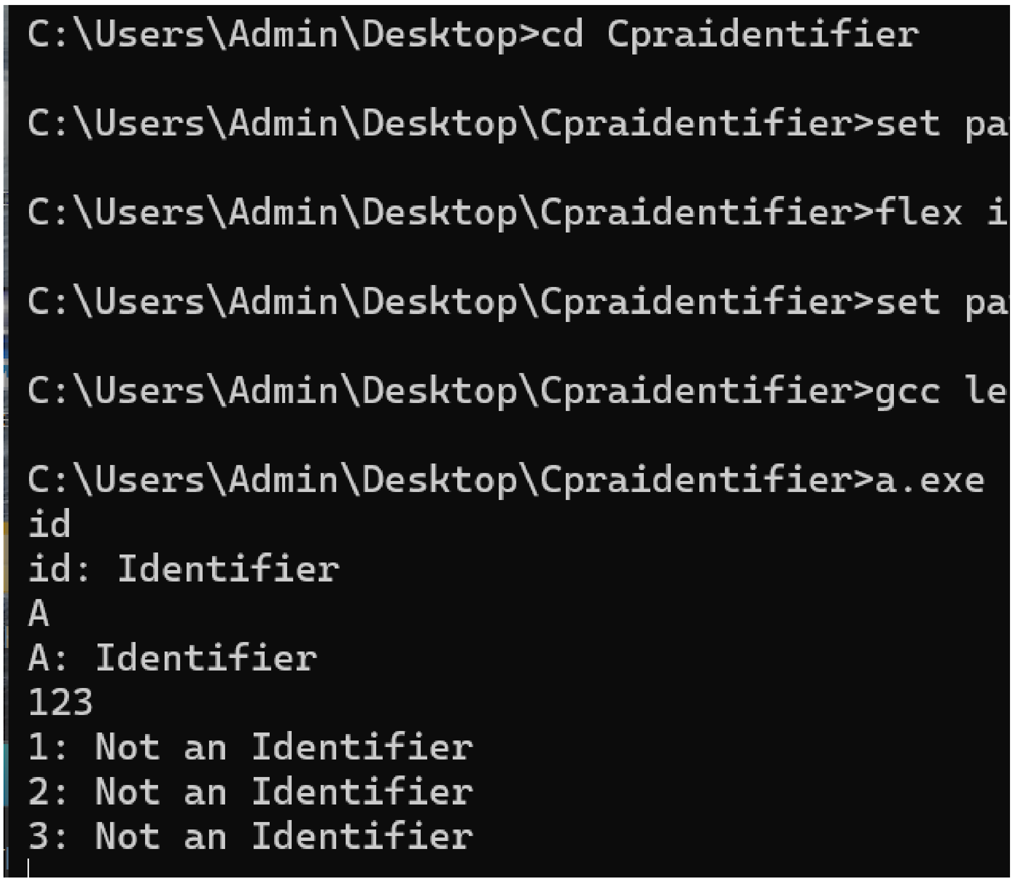
```
C:\Users\Admin\Desktop\Cpraoperatorreco>^Z
C:\Users\Admin\Desktop\Cpraoperatorreco>set path=C:\Program Files

C:\Users\Admin\Desktop\Cpraoperatorreco>flex operatorrecor.l.txt

C:\Users\Admin\Desktop\Cpraoperatorreco>set path=C:\MinGW\bin

C:\Users\Admin\Desktop\Cpraoperatorreco>gcc lex.yy.c

C:\Users\Admin\Desktop\Cpraoperatorreco>a.exe
Enter input: a<b
WORD: a
LESS THAN: <
WORD: b
a==b
WORD: a
EQUALS: ==
WORD: b
```

24. Develop a lexical Analyzer to test whether a given identifier is valid or not.

Program:

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

%%
[a-zA-Z][a-zA-Z0-9_]* { printf("%s: Identifier\n", yytext); }
[^a-zA-Z0-9_ \t\n]+  { printf("%s: Not an Identifier\n", yytext); }
[ \t\n]+
. { printf("%s: Not an Identifier\n", yytext); }
%%

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

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

Output:



```
C:\Users\Admin\Desktop>cd Cpraidentifier

C:\Users\Admin\Desktop\Cpraidentifier>set pa

C:\Users\Admin\Desktop\Cpraidentifier>flex i

C:\Users\Admin\Desktop\Cpraidentifier>set pa

C:\Users\Admin\Desktop\Cpraidentifier>gcc le

C:\Users\Admin\Desktop\Cpraidentifier>a.exe
id
id: Identifier
A
A: Identifier
123
1: Not an Identifier
2: Not an Identifier
3: Not an Identifier
```