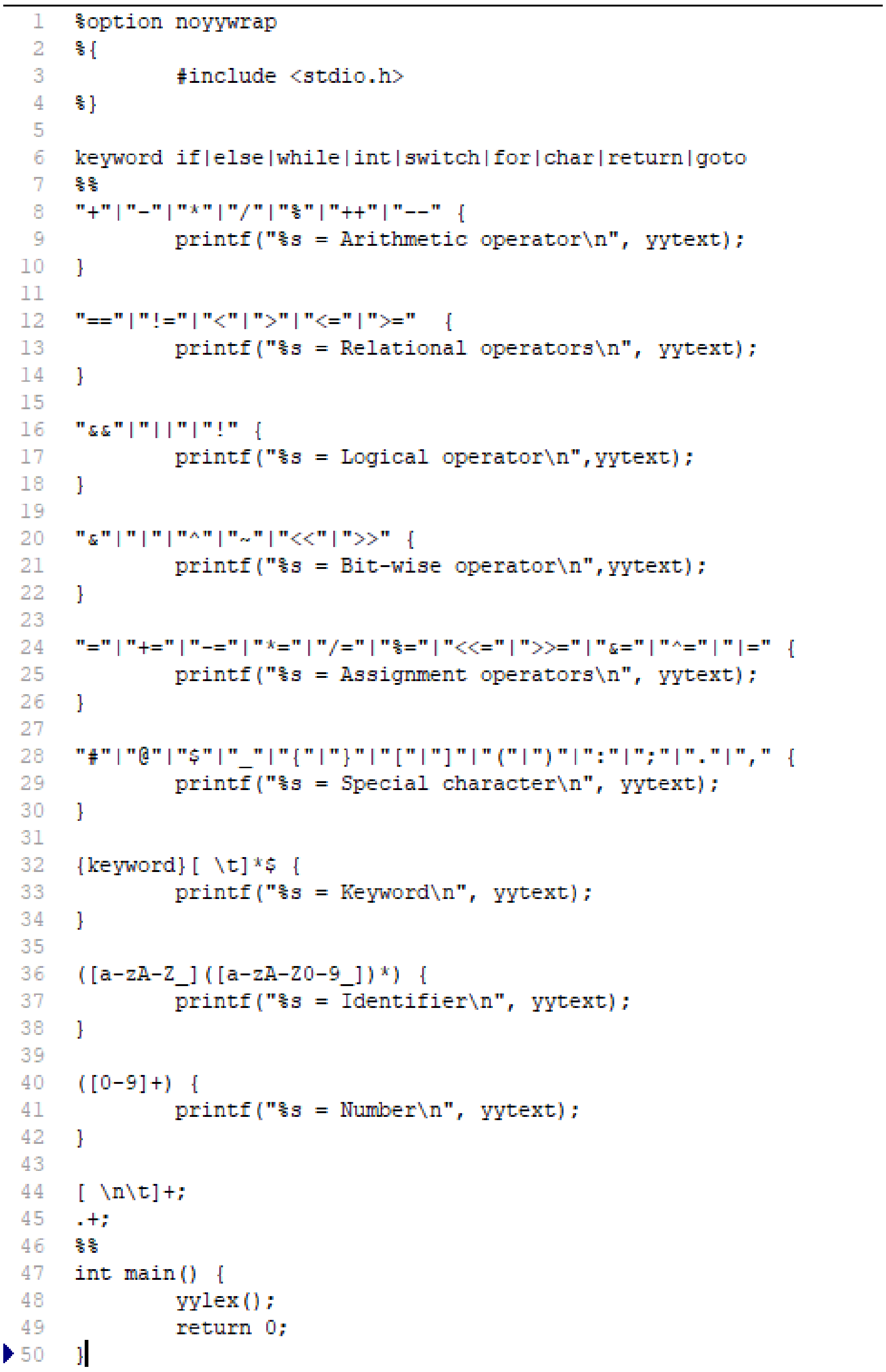
21/09/2021 **CD LAB 3** **2019103573**

EXCERSISES

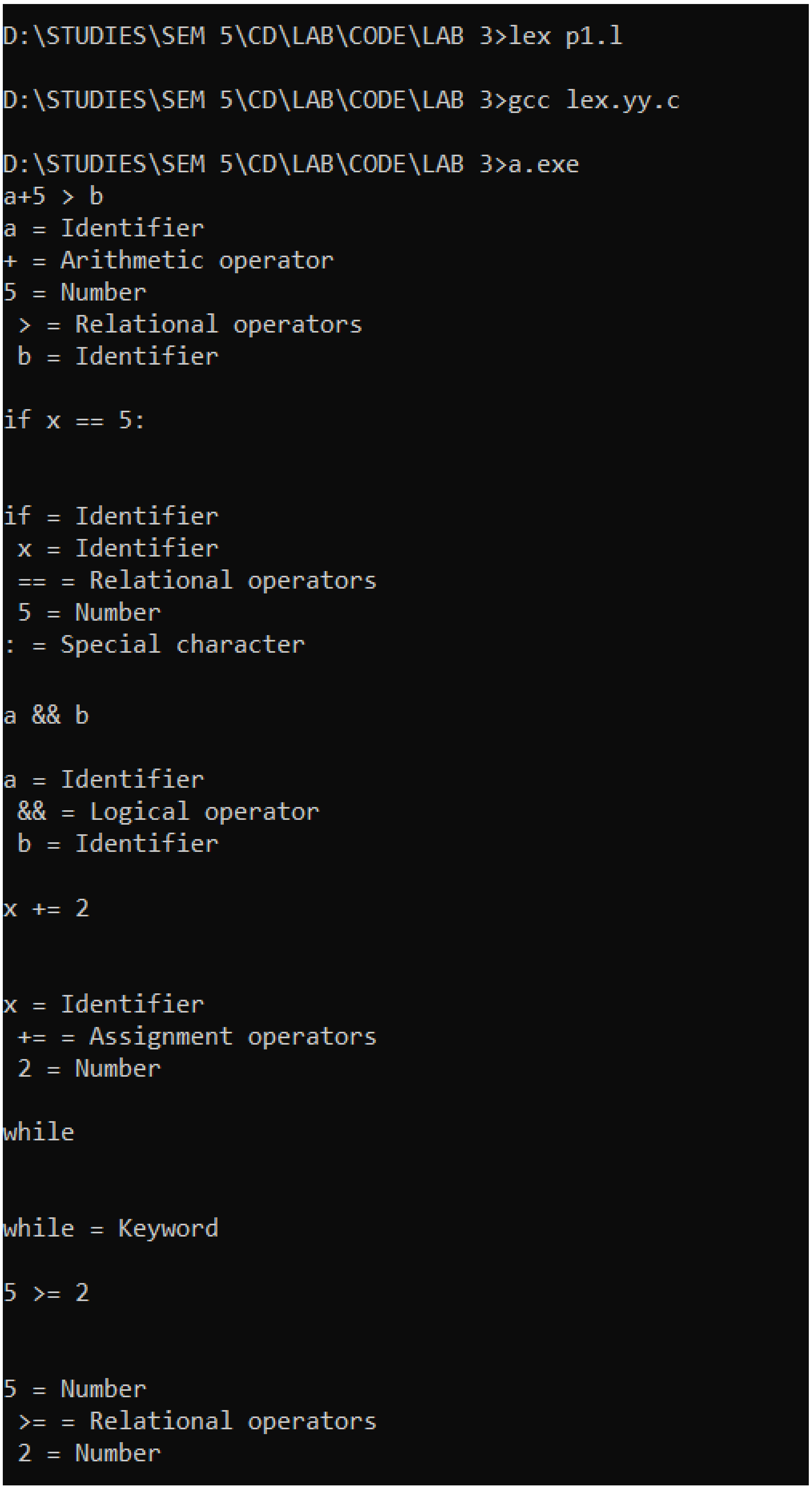
1.Write a lex program to identify whether a given symbol is operator symbol or not and identify its token name.

Code :



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

keyword if|else|while|int|switch|for|char|return|goto
%%
"+"|"-"|"*"|"/"|"%"|"++"|"--" {
	printf("%s = Arithmetic operator\n", yytext);
}

"=="|"!="|"<"|">"|"<="|">="  {
	printf("%s = Relational operators\n", yytext);
}

"&&"|"||"|"!" {
	printf("%s = Logical operator\n",yytext);
}

"&"|"|"|"^"|"~"|"<<"|">>" {
	printf("%s = Bit-wise operator\n",yytext);
}

"="|"+="|"-="|"*="|"/="|"%="|"<<="|">>="|"&="|"^="|"|=" {
	printf("%s = Assignment operators\n", yytext);
}

"#"|"@"|"$"|"_"|"{"|"}"|"["|"]"|"("|")"|":"|";"|"."|"," {
	printf("%s = Special character\n", yytext);
}

{keyword}[ \t]*$ {
	printf("%s = Keyword\n", yytext);
}

([a-zA-Z_]([a-zA-Z0-9_])*) {
	printf("%s = Identifier\n", yytext);
}

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

[ \n\t]+;
.+;
%%
int main() {
	yylex();
	return 0;
}
```

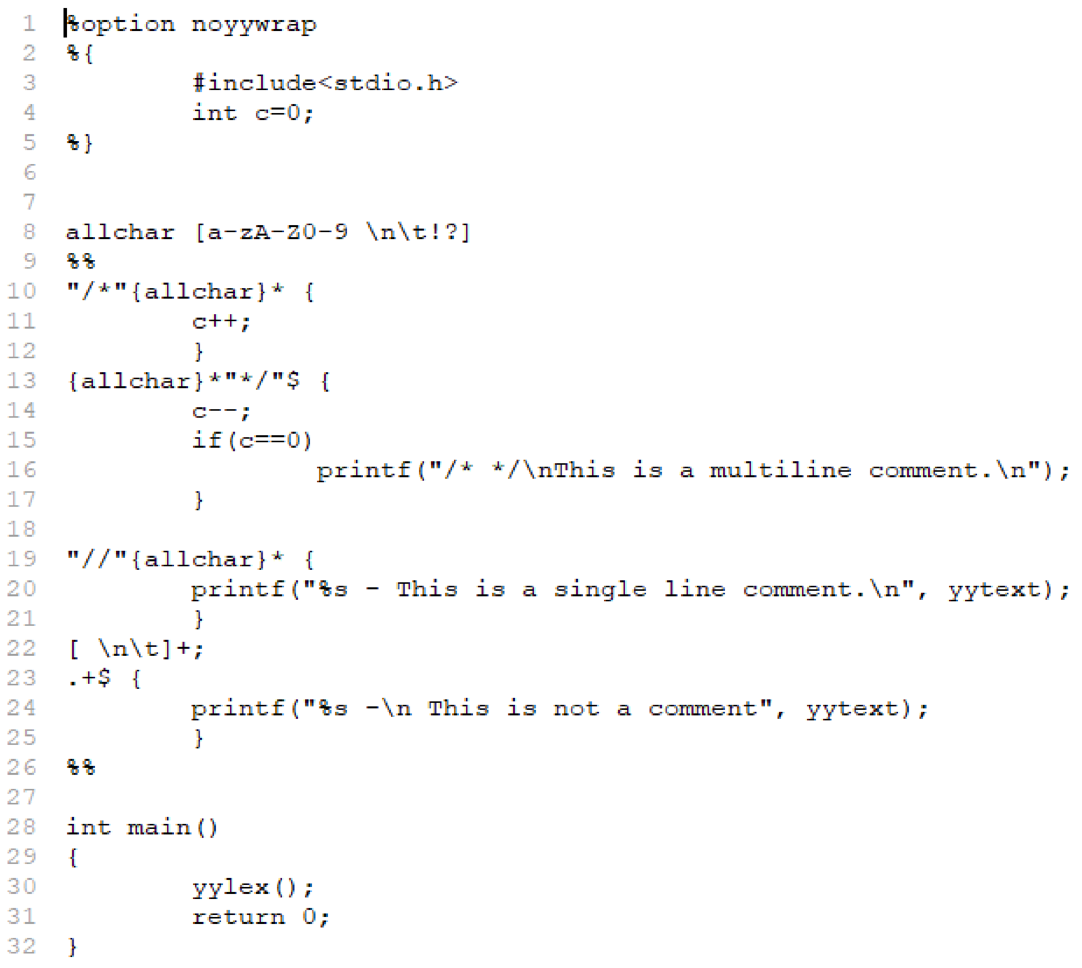
Output :



```
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 3>lex p1.l

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 3>gcc lex.yy.c

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 3>a.exe
a+5 > b
a = Identifier
+ = Arithmetic operator
5 = Number
 > = Relational operators
 b = Identifier

if x == 5:


if = Identifier
 x = Identifier
 == = Relational operators
 5 = Number
: = Special character

a && b

a = Identifier
 && = Logical operator
 b = Identifier

x += 2


x = Identifier
 += = Assignment operators
 2 = Number

while


while = Keyword

5 >= 2


5 = Number
 >= = Relational operators
 2 = Number
```

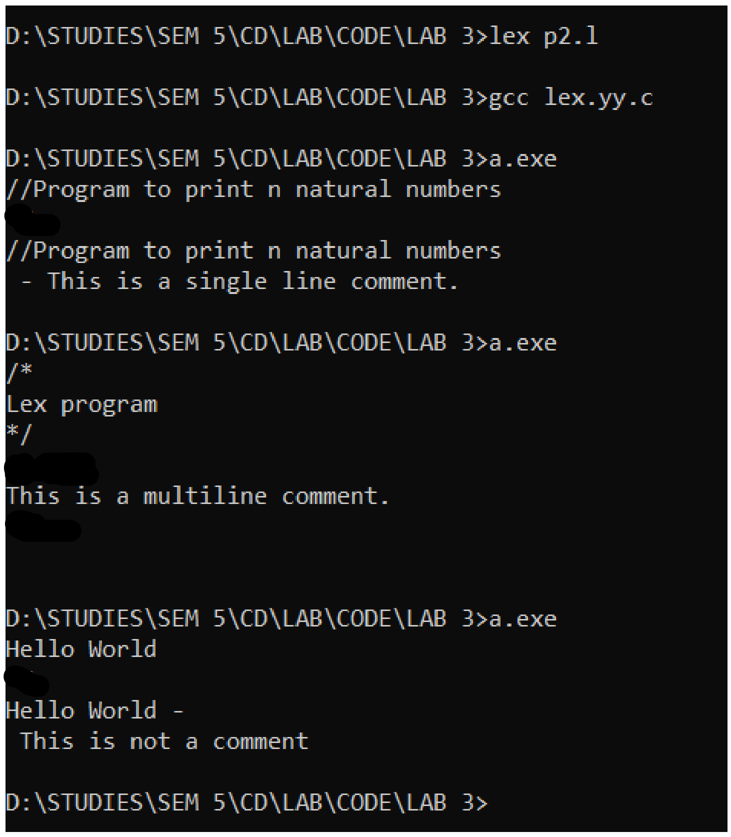
2.Write a lex program to identify whether a given line is a comment or not.

Code :



```
%option noyywrap
%{
        #include<stdio.h>
        int c=0;
%}


allchar [a-zA-Z0-9 \n\t!?]
%%
"/*"{allchar}* {
        c++;
        }
{allchar}*"*/"$ {
        c--;
        if(c==0)
                printf("/* */\nThis is a multiline comment.\n");
        }

"//"{allchar}* {
        printf("%s - This is a single line comment.\n", yytext);
        }
[ \n\t]+;
.+$ {
        printf("%s -\n This is not a comment", yytext);
        }
%%

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

Output :



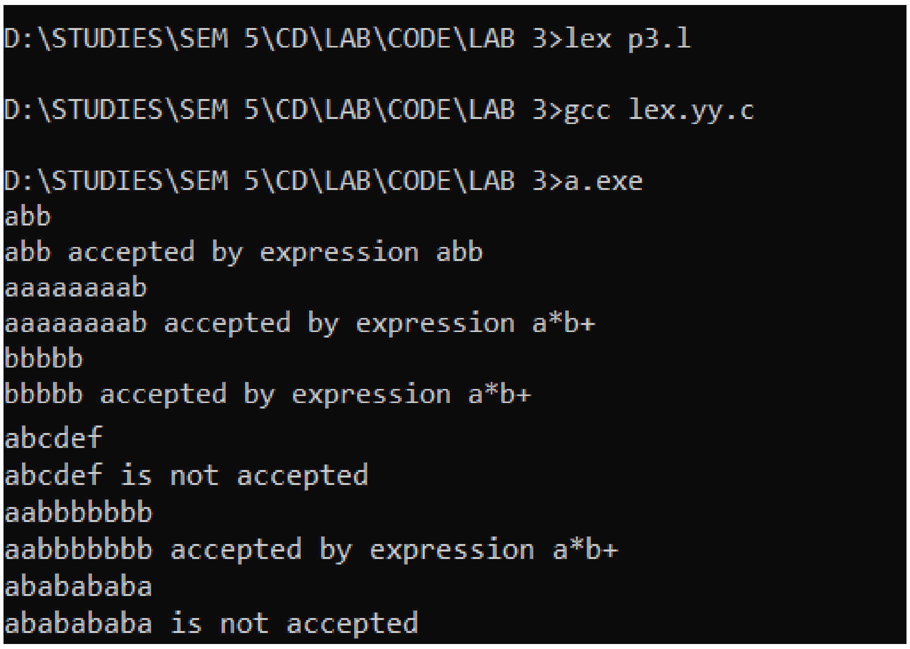
```
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 3>lex p2.l

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 3>gcc lex.yy.c

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 3>a.exe
//Program to print n natural numbers
[redacted]
//Program to print n natural numbers
 - This is a single line comment.

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 3>a.exe
/*
Lex program
*/
[redacted]
This is a multiline comment.
[redacted]


D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 3>a.exe
Hello World
[redacted]
Hello World -
 This is not a comment

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 3>
```

3.Write a lex program to recognize strings under 'a*', 'a*b+', 'abb'.

Code :

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

%%
^(abb)$ {
        printf("%s accepted by expression abb\n", yytext);}
^(a*b+)$ {
        printf("%s accepted by expression a*b+\n", yytext);}
^(a*)[ \n\t]$ {
        printf("%s accepted by expression a*\n", yytext);}
[a-zA-Z0-9]+ {
        printf("%s is not accepted\n", yytext);
}
[\n\t]
.*
%%
int main()
{
        yylex();
        return 0;
}
```

Output :



```
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 3>lex p3.l

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 3>gcc lex.yy.c

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 3>a.exe
abb
abb accepted by expression abb
aaaaaaaab
aaaaaaaab accepted by expression a*b+
bbbbb
bbbbb accepted by expression a*b+
abcdef
abcdef is not accepted
aabbbbbbb
aabbbbbbb accepted by expression a*b+
ababababa
ababababa is not accepted
```