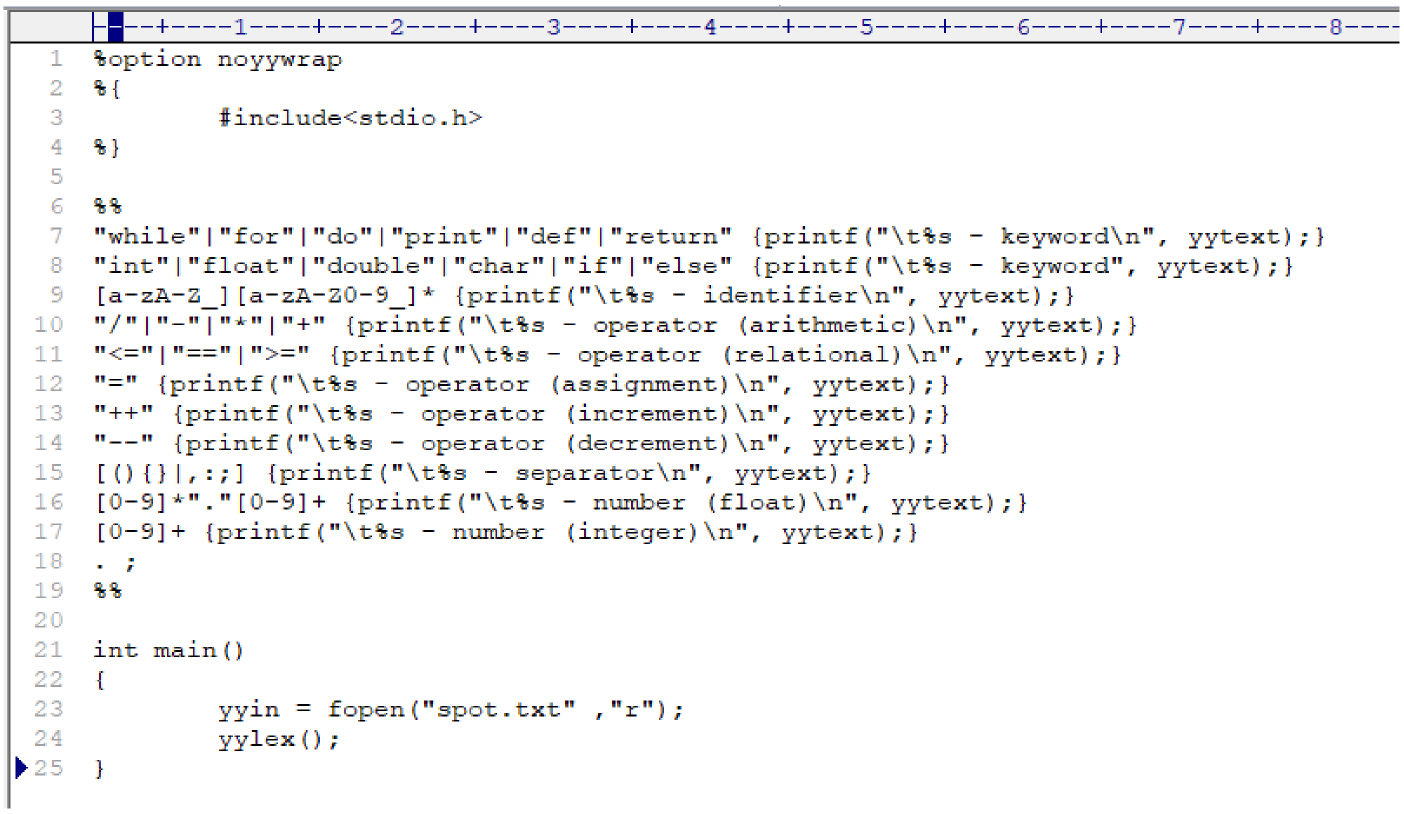
28/09/2021 **CD SPOT 4** 2019103573

SPOT – EXCERSISE

1. Consider the input expressions and identify the tokens in expression

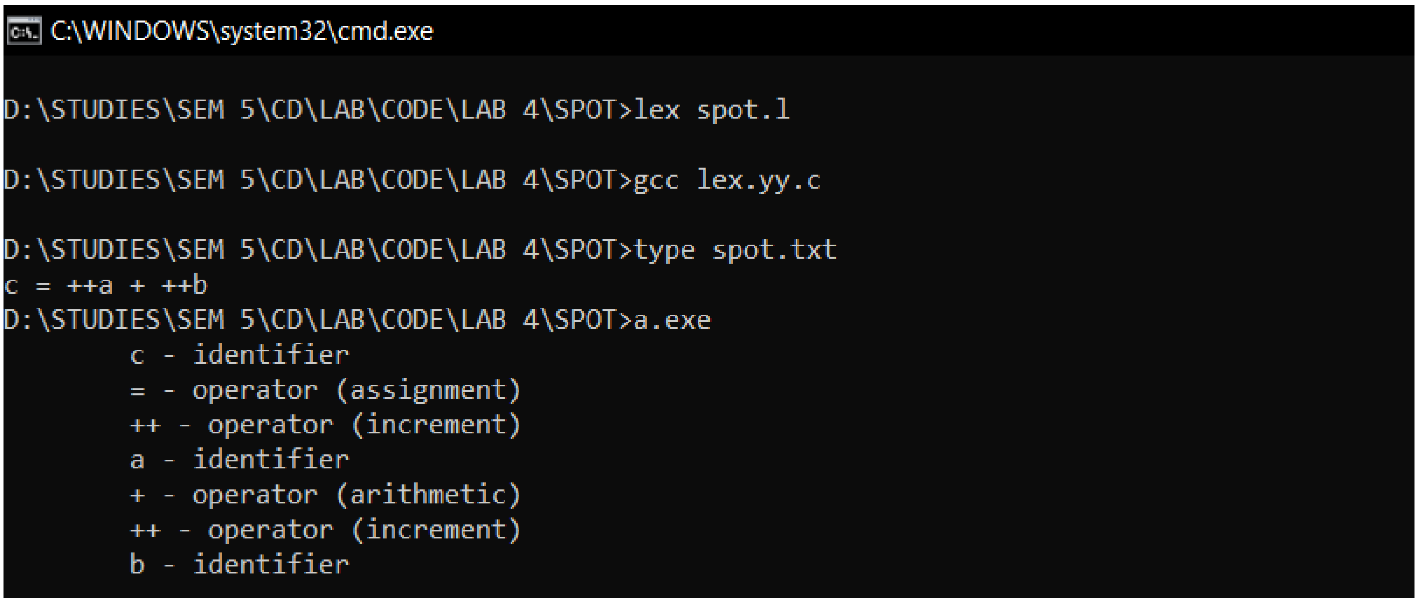
CODE :-



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

%%
"while"|"for"|"do"|"print"|"def"|"return" {printf("\t%s - keyword\n", yytext);}
"int"|"float"|"double"|"char"|"if"|"else" {printf("\t%s - keyword", yytext);}
[a-zA-Z_][a-zA-Z0-9_]* {printf("\t%s - identifier\n", yytext);}
"/"|"-"|"*"|"+" {printf("\t%s - operator (arithmetic)\n", yytext);}
"<="|"=="|">=" {printf("\t%s - operator (relational)\n", yytext);}
"=" {printf("\t%s - operator (assignment)\n", yytext);}
"++" {printf("\t%s - operator (increment)\n", yytext);}
"--" {printf("\t%s - operator (decrement)\n", yytext);}
[(){}|,:;] {printf("\t%s - separator\n", yytext);}
[0-9]*"."[0-9]+ {printf("\t%s - number (float)\n", yytext);}
[0-9]+ {printf("\t%s - number (integer)\n", yytext);}
. ;
%%

int main()
{
        yyin = fopen("spot.txt" ,"r");
        yylex();
}
```

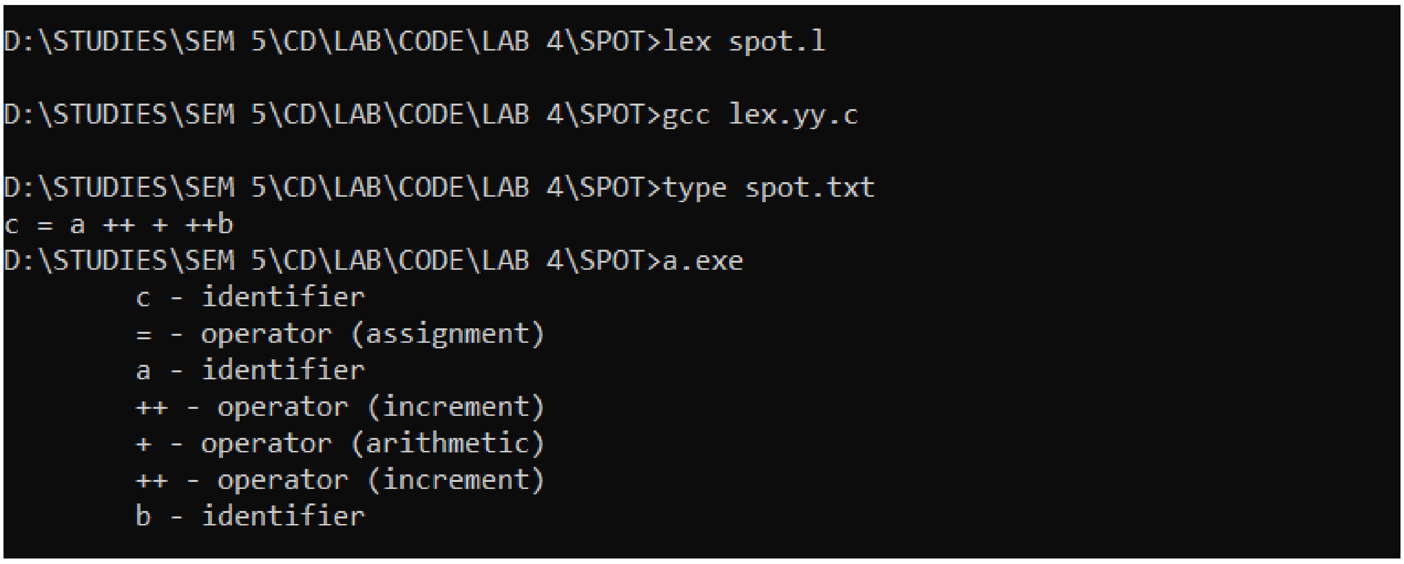
OUTPUT :-

a. c = ++a + ++b



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

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>lex spot.l

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>gcc lex.yy.c

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>type spot.txt
c = ++a + ++b
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>a.exe
        c - identifier
        = - operator (assignment)
        ++ - operator (increment)
        a - identifier
        + - operator (arithmetic)
        ++ - operator (increment)
        b - identifier
```

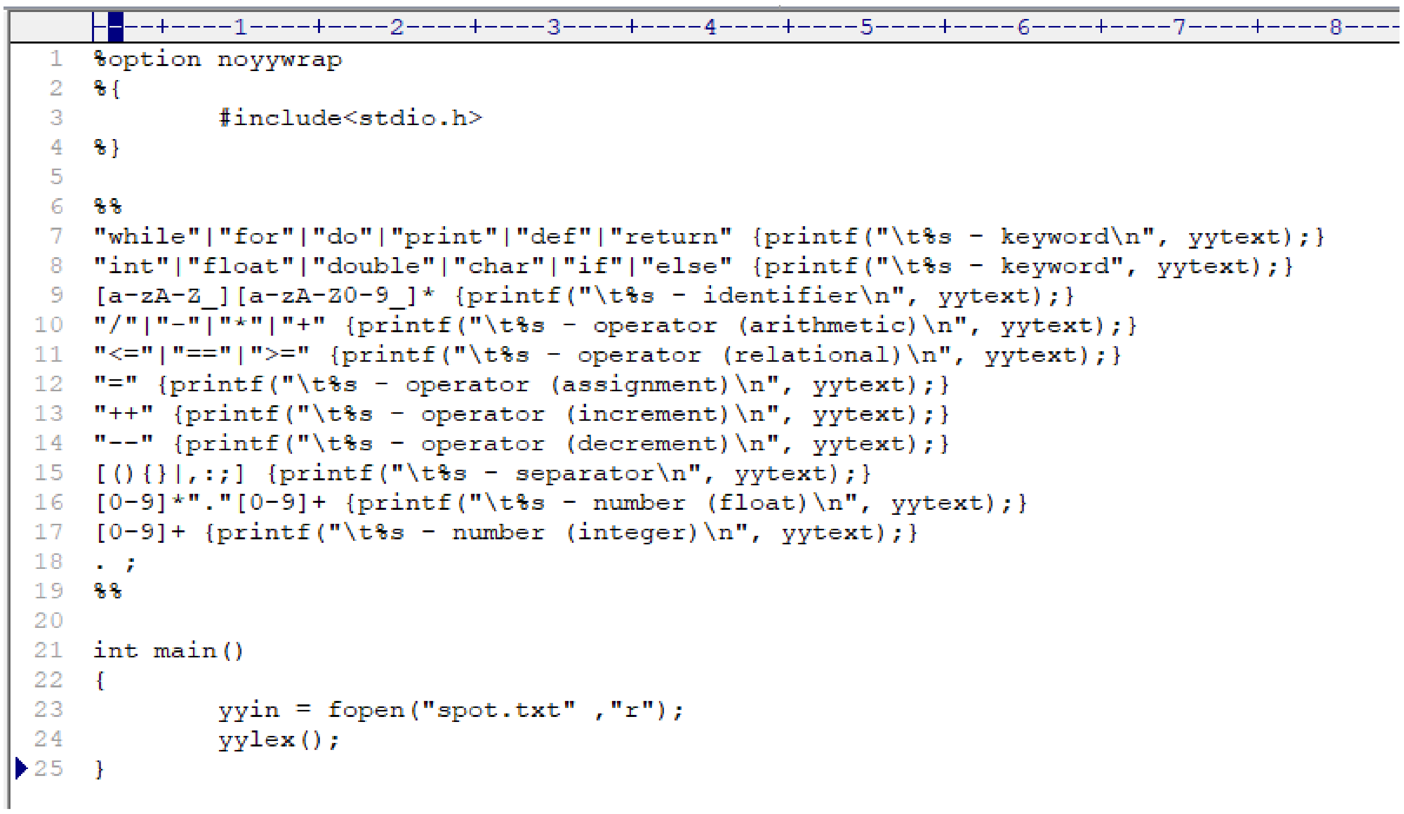
b. c = a ++ + ++b



```
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>lex spot.l

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>gcc lex.yy.c

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>type spot.txt
c = a ++ + ++b
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>a.exe
        c - identifier
        = - operator (assignment)
        a - identifier
        ++ - operator (increment)
        + - operator (arithmetic)
        ++ - operator (increment)
        b - identifier
```

2. Identify the tokens in the given input statement

CODE :-



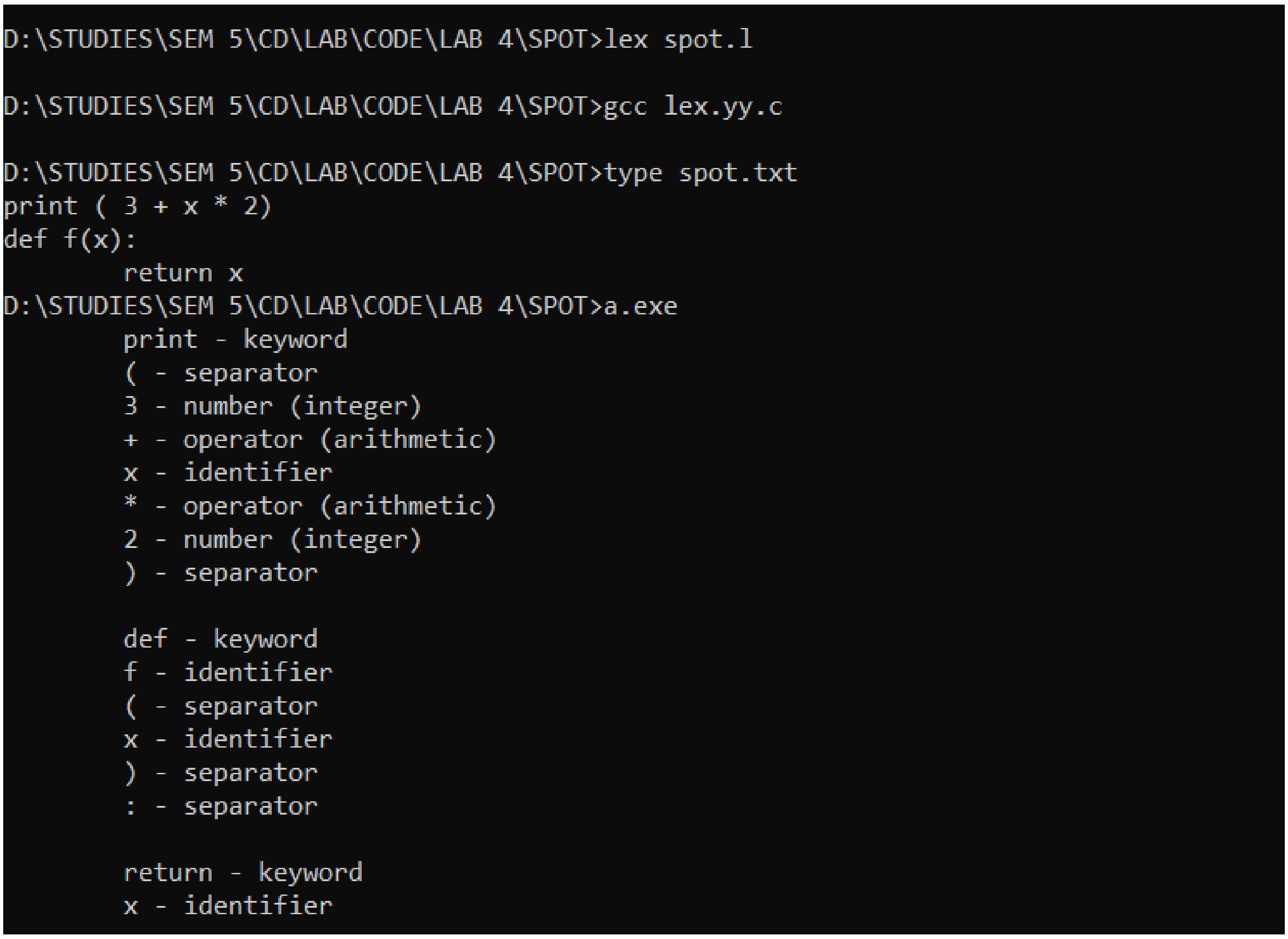
```
%option noyywrap
%{
        #include<stdio.h>
%}

%%
"while"|"for"|"do"|"print"|"def"|"return" {printf("\t%s - keyword\n", yytext);}
"int"|"float"|"double"|"char"|"if"|"else" {printf("\t%s - keyword", yytext);}
[a-zA-Z_][a-zA-Z0-9_]* {printf("\t%s - identifier\n", yytext);}
"/"|"-"|"*"|"+" {printf("\t%s - operator (arithmetic)\n", yytext);}
"<="|"=="|">=" {printf("\t%s - operator (relational)\n", yytext);}
"=" {printf("\t%s - operator (assignment)\n", yytext);}
"++" {printf("\t%s - operator (increment)\n", yytext);}
"--" {printf("\t%s - operator (decrement)\n", yytext);}
[(){}|,:;] {printf("\t%s - separator\n", yytext);}
[0-9]*"."[0-9]+ {printf("\t%s - number (float)\n", yytext);}
[0-9]+ {printf("\t%s - number (integer)\n", yytext);}
. ;
%%

int main()
{
        yyin = fopen("spot.txt" ,"r");
        yylex();
}
```

OUTPUT :-

a. print (3 + x * 2)

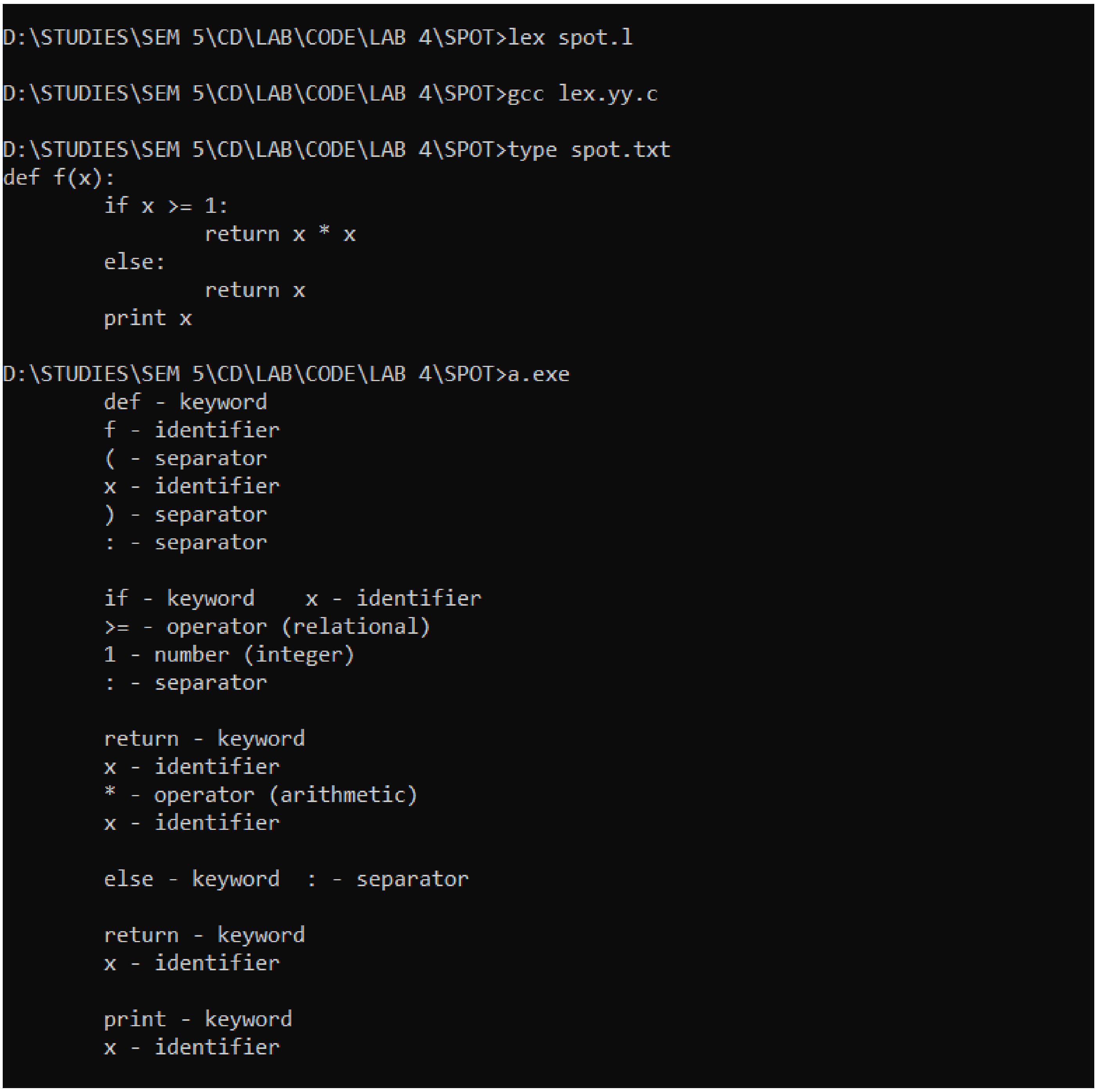
def f(x):

return x



```
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>lex spot.l

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>gcc lex.yy.c

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>type spot.txt
print ( 3 + x * 2)
def f(x):
        return x
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>a.exe
        print - keyword
        ( - separator
        3 - number (integer)
        + - operator (arithmetic)
        x - identifier
        * - operator (arithmetic)
        2 - number (integer)
        ) - separator

        def - keyword
        f - identifier
        ( - separator
        x - identifier
        ) - separator
        : - separator

        return - keyword
        x - identifier
```

```
b. def f(x):
        if x >= 1:
                return x * x
        else:
                return x
        print x
```



```
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>lex spot.l

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>gcc lex.yy.c

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>type spot.txt
def f(x):
        if x >= 1:
                return x * x
        else:
                return x
        print x

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 4\SPOT>a.exe
        def - keyword
        f - identifier
        ( - separator
        x - identifier
        ) - separator
        : - separator

        if - keyword    x - identifier
        >= - operator (relational)
        1 - number (integer)
        : - separator

        return - keyword
        x - identifier
        * - operator (arithmetic)
        x - identifier

        else - keyword  : - separator

        return - keyword
        x - identifier

        print - keyword
        x - identifier
```