02-12-2021 **2019103573**

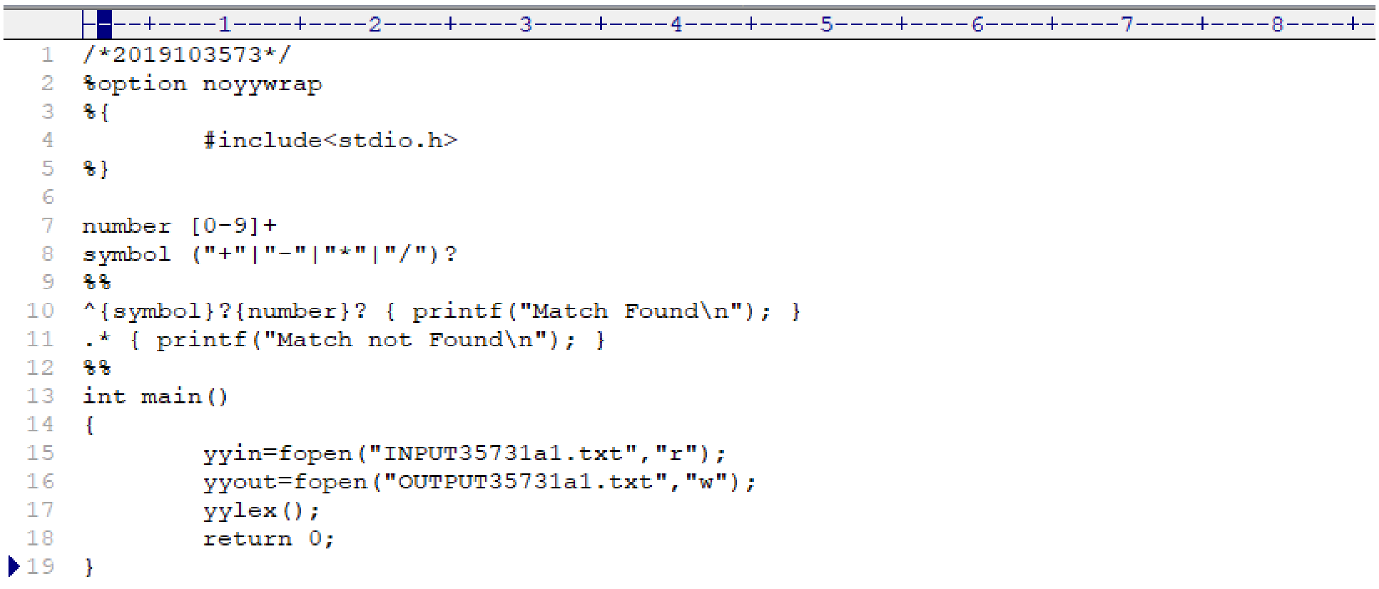
CS6109 – COMPILER DESIGN LABORATORY

LAB ASSESSMENT-SET B

1. a. Write regular definition to display the line of string for the following using LEX.

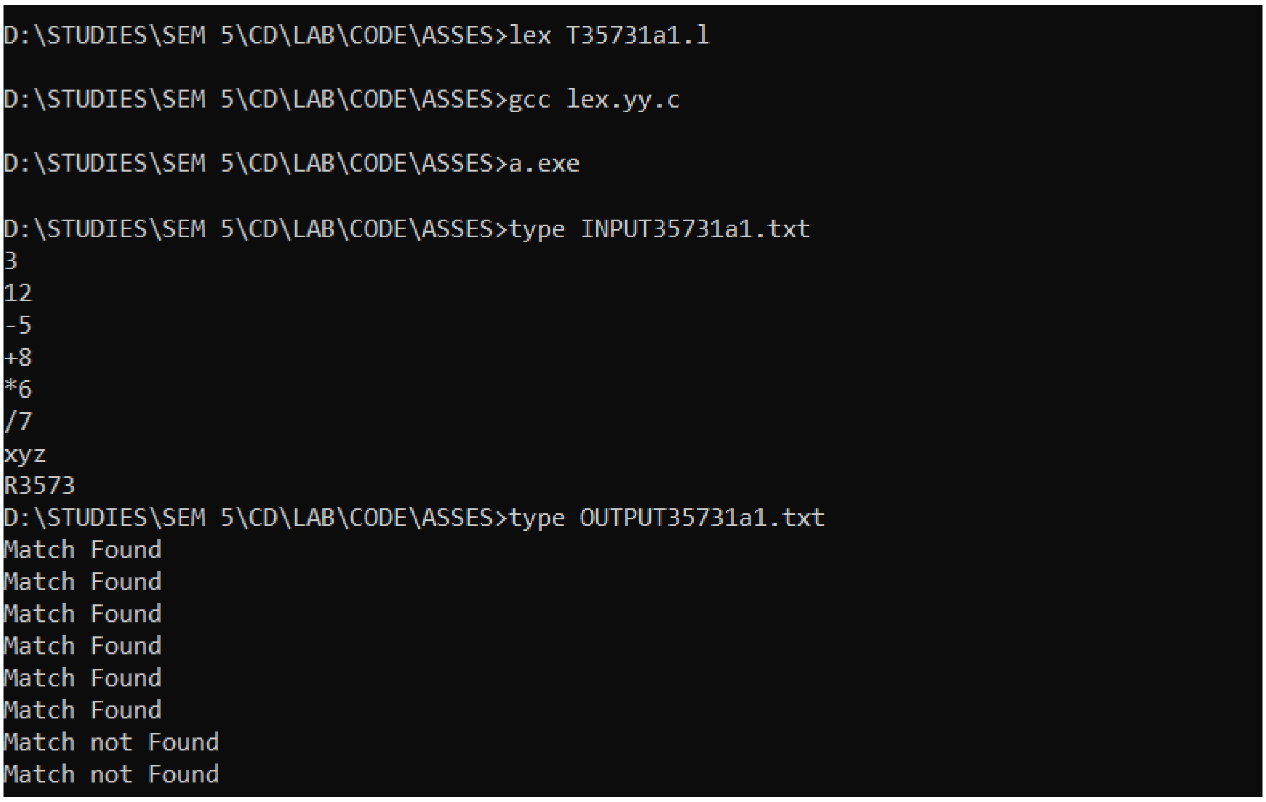
i. Match any string of one or more digits with an optional prefix of +, -, * and /.

LEX



```
/*2019103573*/
%option noyywrap
%{
        #include<stdio.h>
%}

number [0-9]+
symbol ("+"|"-"|"*"|"/")?
%%
^{symbol}?{number}? { printf("Match Found\n"); }
.* { printf("Match not Found\n"); }
%%
int main()
{
        yyin=fopen("INPUT35731a1.txt","r");
        yyout=fopen("OUTPUT35731a1.txt","w");
        yylex();
        return 0;
}
```

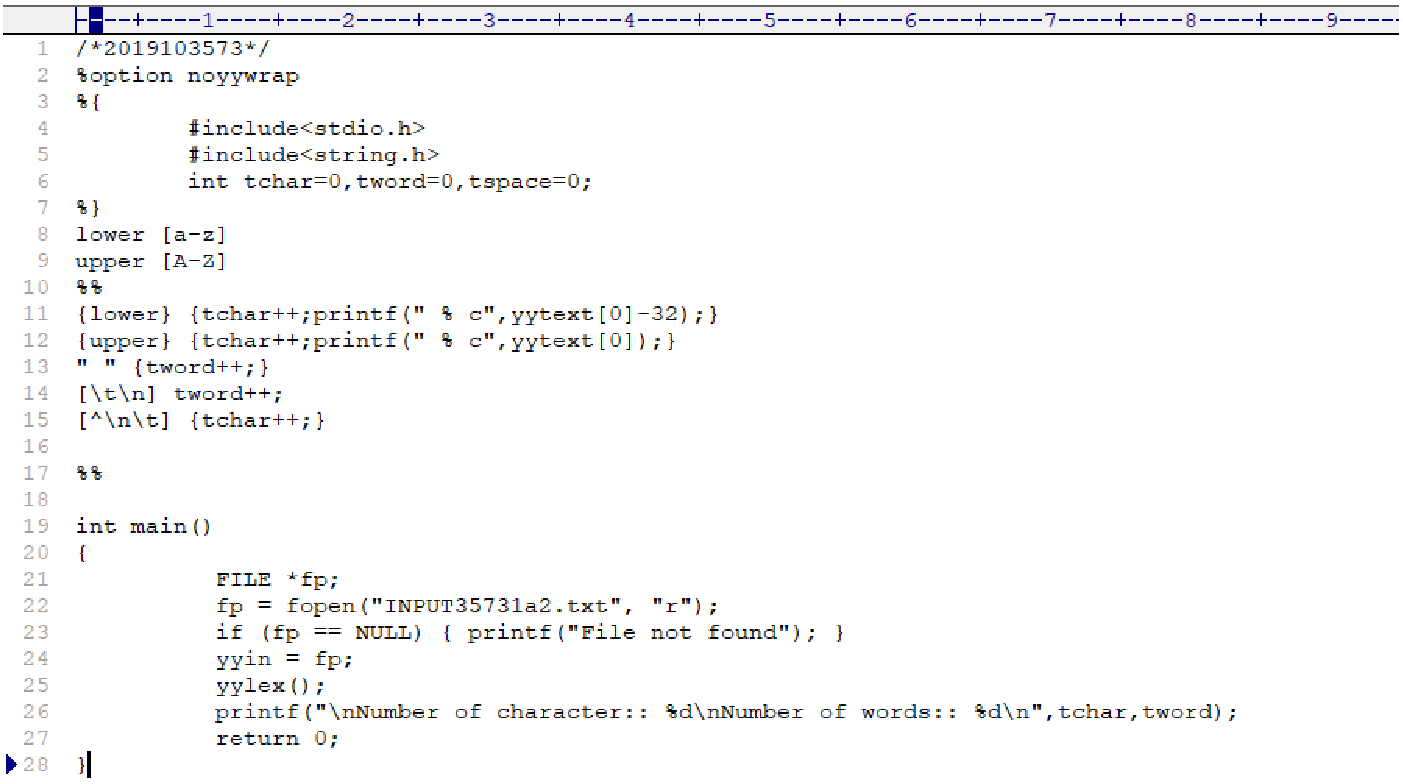
OUTPUT



```
D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>lex T35731a1.l

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

D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>a.exe

D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>type INPUT35731a1.txt
3
12
-5
+8
*6
/7
xyz
R3573
D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>type OUTPUT35731a1.txt
Match Found
Match Found
Match Found
Match Found
Match Found
Match Found
Match not Found
Match not Found
```

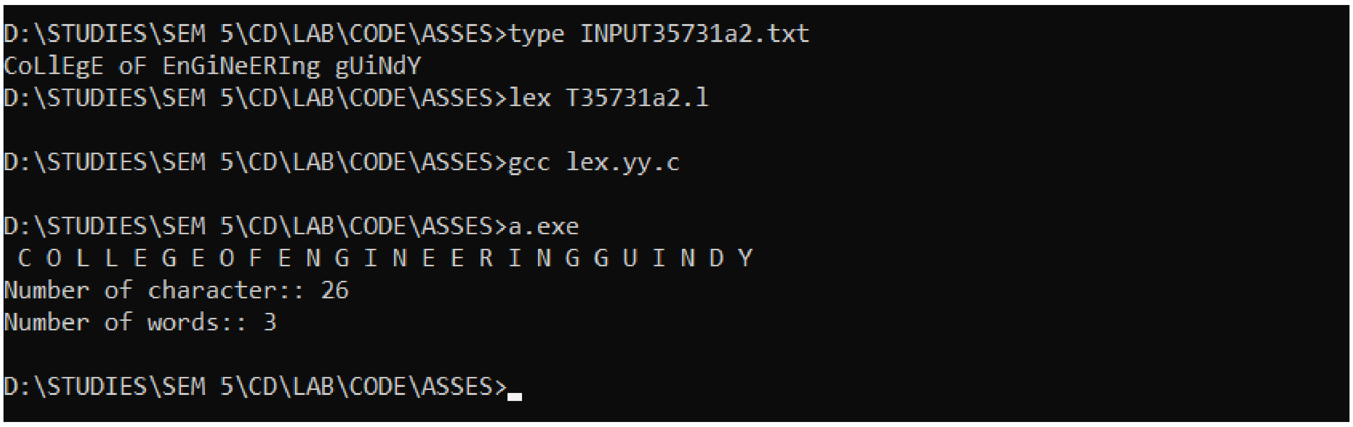
ii. Translating all input string into uppercase, find the character and word count of the input string

LEX



```
/*2019103573*/
%option noyywrap
%{
        #include<stdio.h>
        #include<string.h>
        int tchar=0,tword=0,tspace=0;
%}
lower [a-z]
upper [A-Z]
%%
{lower} {tchar++;printf(" % c",yytext[0]-32);}
{upper} {tchar++;printf(" % c",yytext[0]);}
" " {tword++;}
[\t\n] tword++;
[^\n\t] {tchar++;}

%%

int main()
{
          FILE *fp;
          fp = fopen("INPUT35731a2.txt", "r");
          if (fp == NULL) { printf("File not found"); }
          yyin = fp;
          yylex();
          printf("\nNumber of character:: %d\nNumber of words:: %d\n",tchar,tword);
          return 0;
}
```

OUTPUT

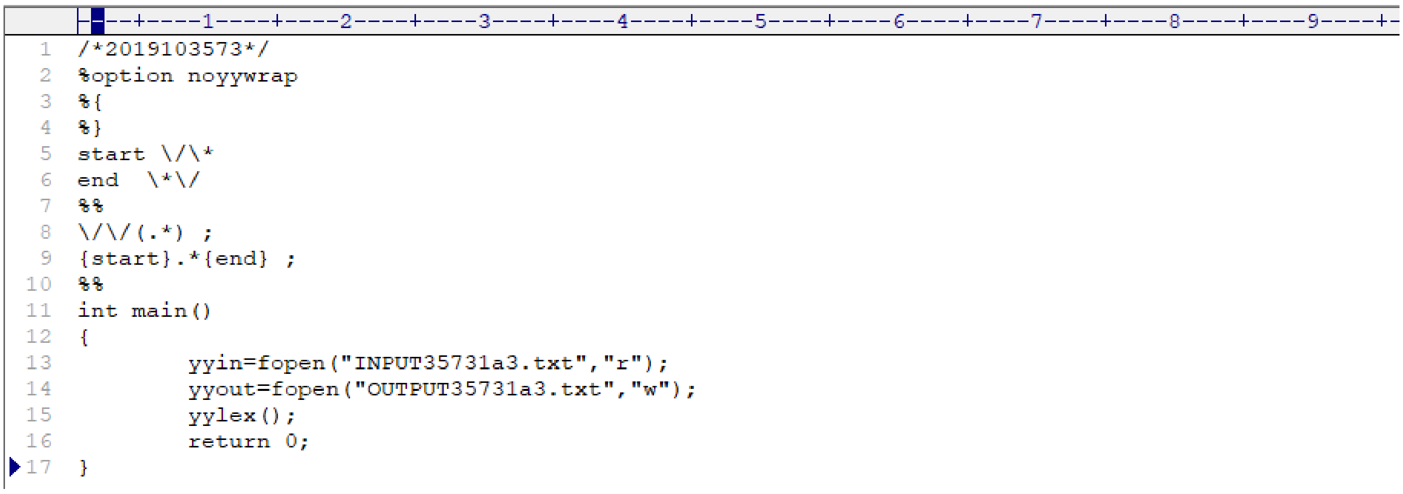


```
D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>type INPUT35731a2.txt
CoLlEgE oF EnGiNeERIng gUiNdY
D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>lex T35731a2.l

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

D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>a.exe
 C O L L E G E O F E N G I N E E R I N G G U I N D Y
Number of character:: 26
Number of words:: 3

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

iii. Eliminating all C-like comments from a text file

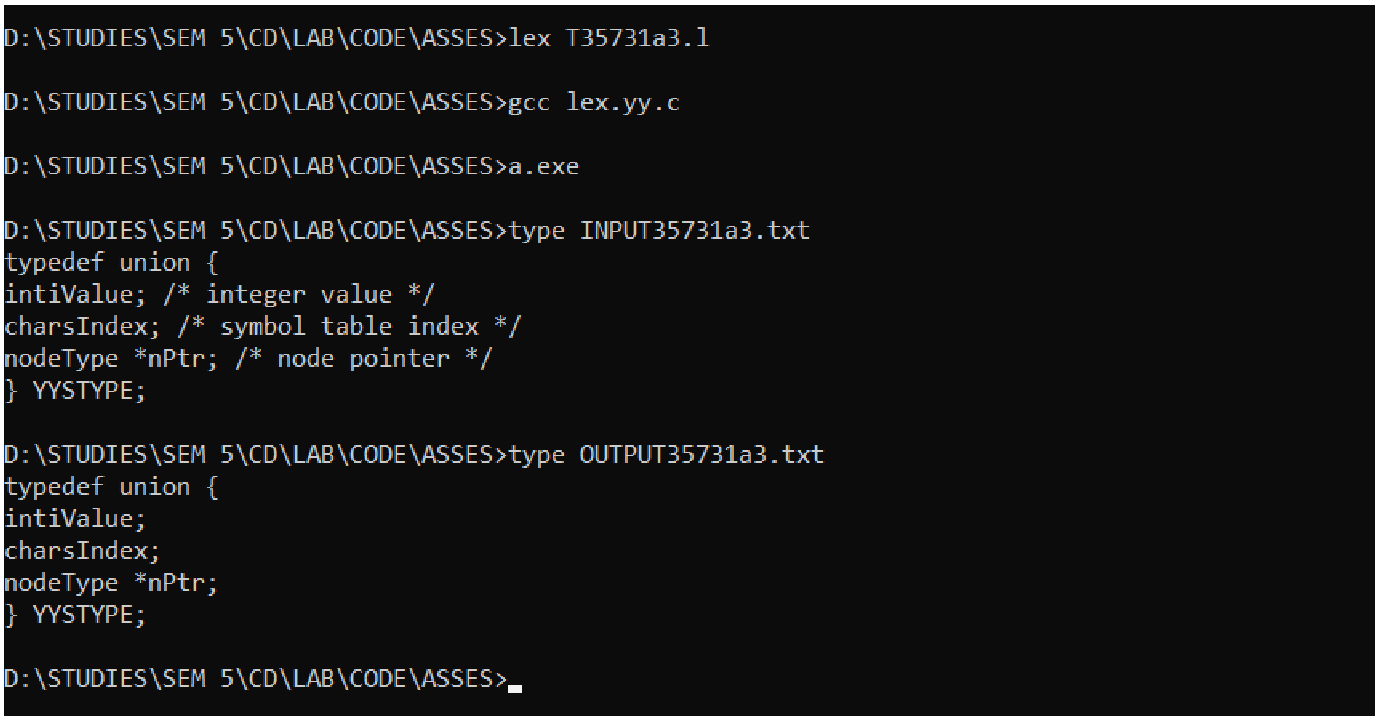
```
typedef union {
intiValue; /* integer value */
charsIndex; /* symbol table index */
nodeType *nPtr; /* node pointer */
} YYSTYPE;
```

LEX



```
/*2019103573*/
%option noyywrap
%{
%}
start \/\*
end  \*\/
%%
\/\/(.*) ;
{start}.*{end} ;
%%
int main()
{
        yyin=fopen("INPUT35731a3.txt","r");
        yyout=fopen("OUTPUT35731a3.txt","w");
        yylex();
        return 0;
}
```

OUTPUT

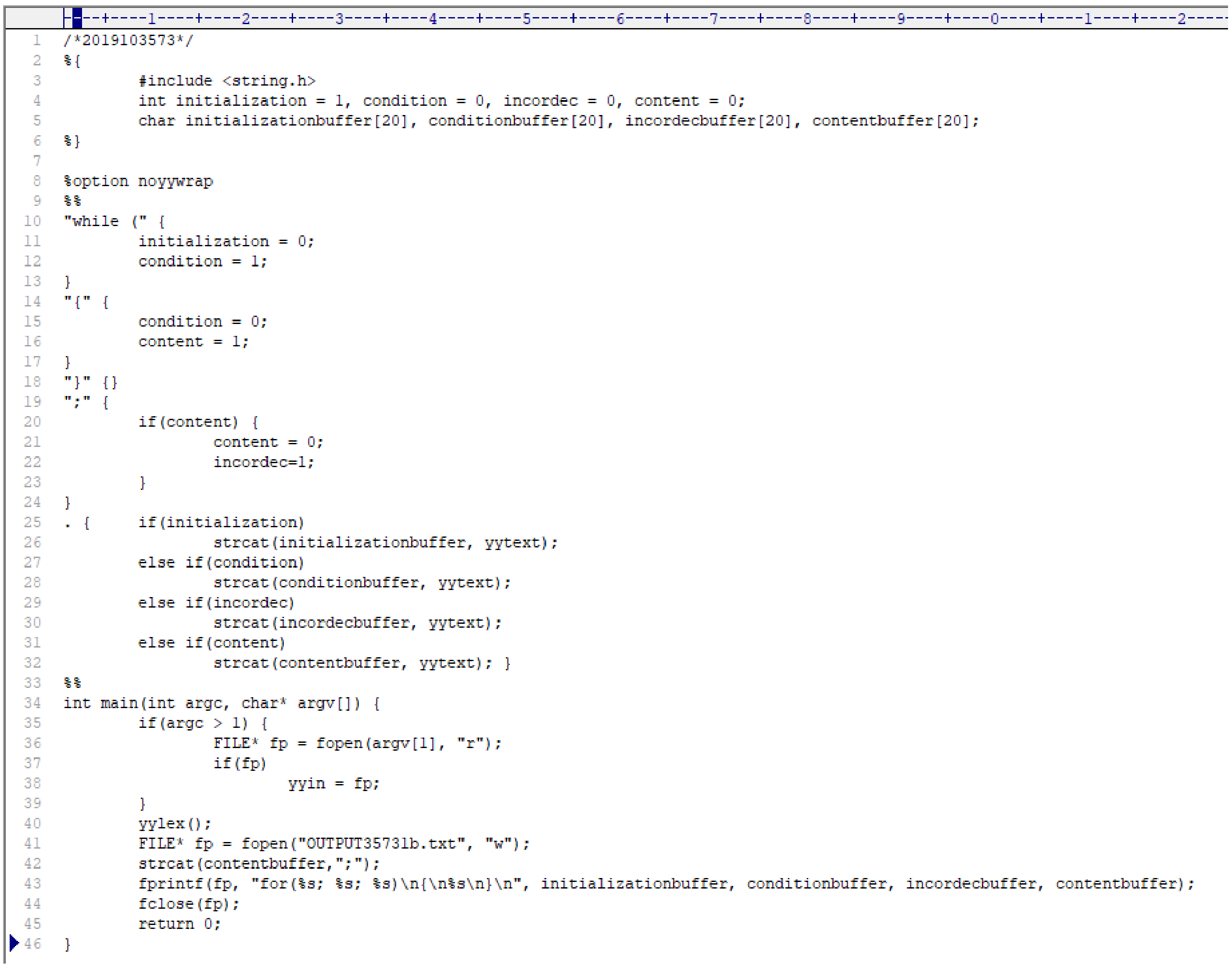


```
D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>lex T35731a3.l

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

D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>a.exe

D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>type INPUT35731a3.txt
typedef union {
intiValue; /* integer value */
charsIndex; /* symbol table index */
nodeType *nPtr; /* node pointer */
} YYSTYPE;

D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>type OUTPUT35731a3.txt
typedef union {
intiValue;
charsIndex;
nodeType *nPtr;
} YYSTYPE;

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

b. Convert the while loop to nested for statement

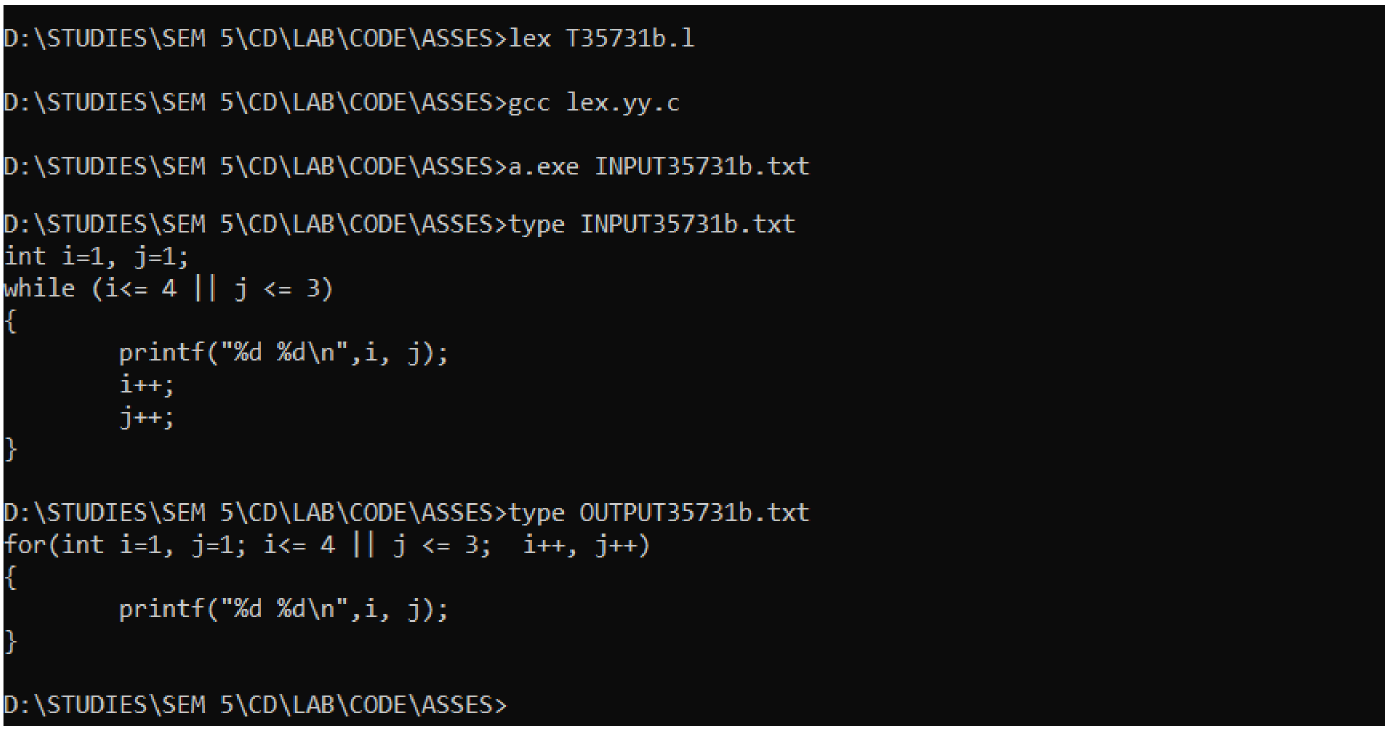
```
int i=1, j=1;
while (i<= 4 || j <= 3)
{
        printf("%d %d\n",i, j);
        i++;
        j++;
}
```

LEX



```
/*2019103573*/
%{
        #include <string.h>
        int initialization = 1, condition = 0, incordec = 0, content = 0;
        char initializationbuffer[20], conditionbuffer[20], incordecbuffer[20], contentbuffer[20];
%}

%option noyywrap
%%
"while (" {
        initialization = 0;
        condition = 1;
}
"{" {
        condition = 0;
        content = 1;
}
"}" {}
";" {
        if(content) {
                content = 0;
                incordec=1;
        }
}
. {     if(initialization)
                strcat(initializationbuffer, yytext);
        else if(condition)
                strcat(conditionbuffer, yytext);
        else if(incordec)
                strcat(incordecbuffer, yytext);
        else if(content)
                strcat(contentbuffer, yytext); }
%%
int main(int argc, char* argv[]) {
        if(argc > 1) {
                FILE* fp = fopen(argv[1], "r");
                if(fp)
                        yyin = fp;
        }
        yylex();
        FILE* fp = fopen("OUTPUT35731b.txt", "w");
        strcat(contentbuffer,";");
        fprintf(fp, "for(%s; %s; %s)\n{\n%s\n}\n", initializationbuffer, conditionbuffer, incordecbuffer, contentbuffer);
        fclose(fp);
        return 0;
}
```

OUTPUT



```
D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>lex T35731b.l

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

D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>a.exe INPUT35731b.txt

D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>type INPUT35731b.txt
int i=1, j=1;
while (i<= 4 || j <= 3)
{
        printf("%d %d\n",i, j);
        i++;
        j++;
}

D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>type OUTPUT35731b.txt
for(int i=1, j=1; i<= 4 || j <= 3;  i++, j++)
{
        printf("%d %d\n",i, j);
}

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

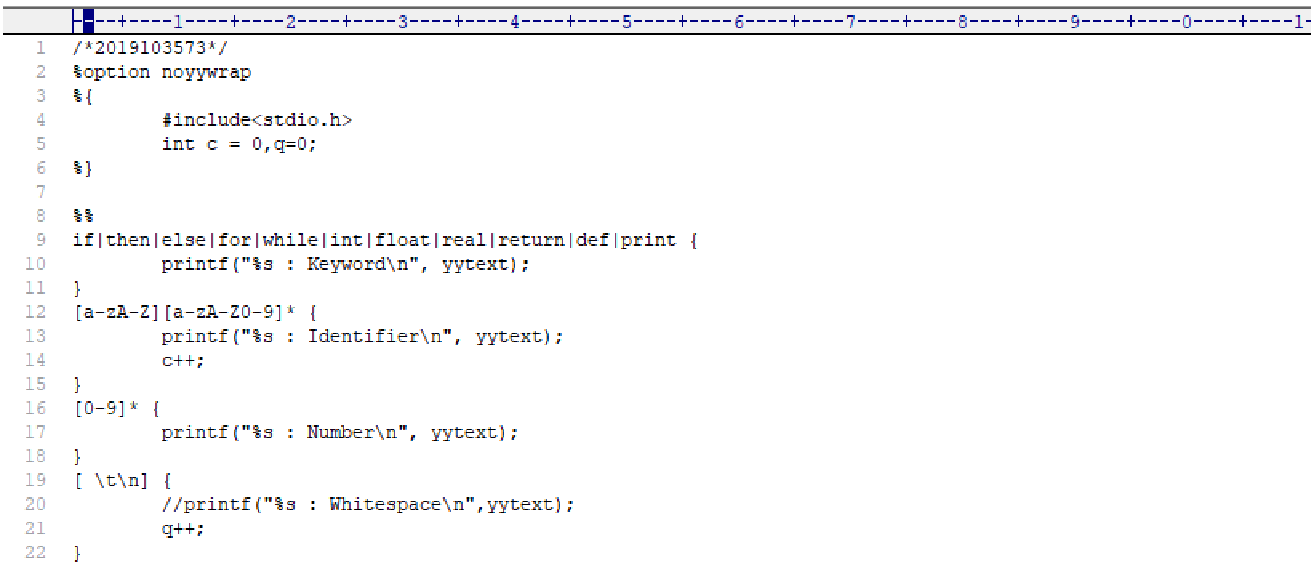
2. Consider the following program fragment

```
      inti, j, a[2][3] ;
      float c , x;
      for ( i = 1; i<= 10 ; i++){
            for ( j = 1 ; j <=10 ; j ++ ){
                  a[i][j] = 1;
                  x = c + a[i][j] ;
            }
      }
```

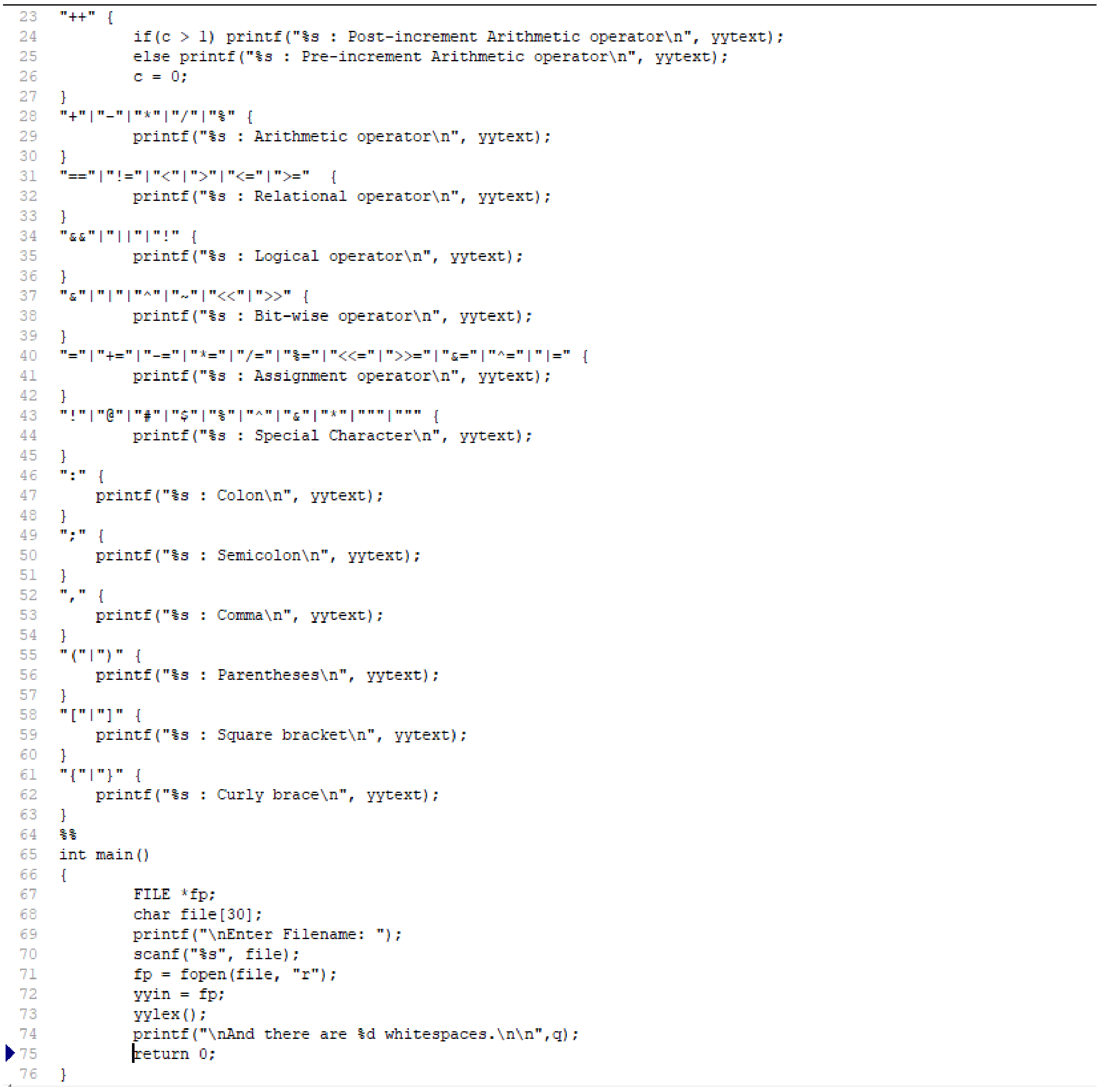
Perform the following using LEX/YACC

a. Identify the tokens and print them

LEX

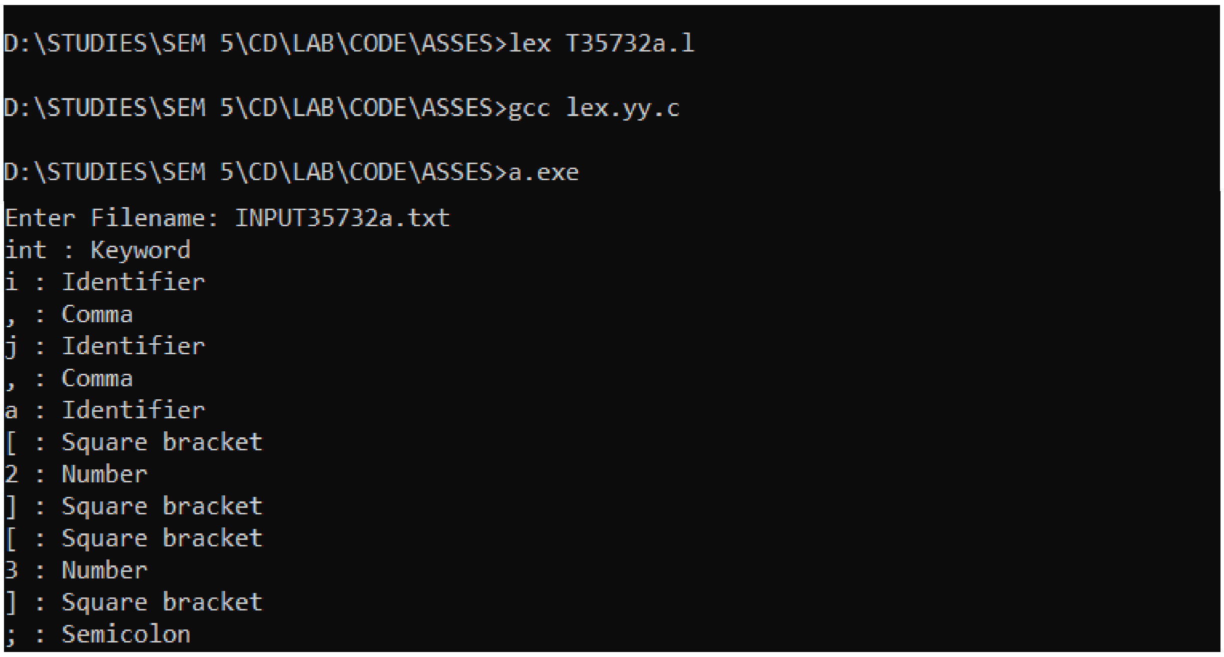


```
/*2019103573*/
%option noyywrap
%{
        #include<stdio.h>
        int c = 0,q=0;
%}

%%
if|then|else|for|while|int|float|real|return|def|print {
        printf("%s : Keyword\n", yytext);
}
[a-zA-Z][a-zA-Z0-9]* {
        printf("%s : Identifier\n", yytext);
        c++;
}
[0-9]* {
        printf("%s : Number\n", yytext);
}
[ \t\n] {
        //printf("%s : Whitespace\n",yytext);
        q++;
}
```

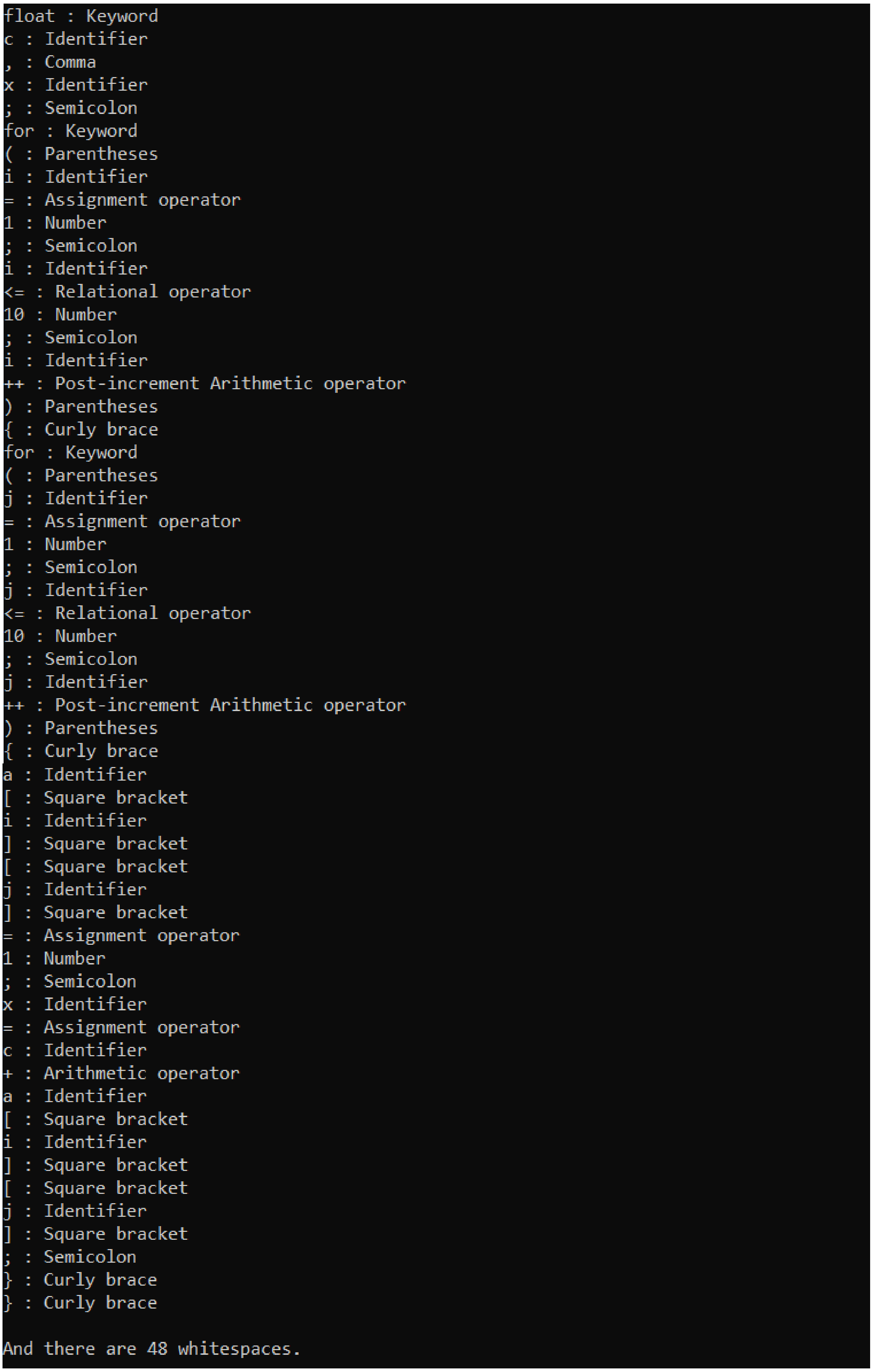


```
"++" {
        if(c > 1) printf("%s : Post-increment Arithmetic operator\n", yytext);
        else printf("%s : Pre-increment Arithmetic operator\n", yytext);
        c = 0;
}
"+"|"-"|"*"|"/"|"%" {
        printf("%s : Arithmetic operator\n", yytext);
}
"=="|"!="|"<"|">"|"<="|">="  {
        printf("%s : Relational operator\n", yytext);
}
"&&"|"||"|"!" {
        printf("%s : Logical operator\n", yytext);
}
"&"|"|"|"^"|"~"|"<<"|">>" {
        printf("%s : Bit-wise operator\n", yytext);
}
"="|"+="|"-="|"*="|"/="|"%="|"<<="|">>="|"&="|"^="|"|=" {
        printf("%s : Assignment operator\n", yytext);
}
"!"|"@"|"#"|"$"|"%"|"^"|"&"|"*"|"""|""" {
        printf("%s : Special Character\n", yytext);
}
":" {
    printf("%s : Colon\n", yytext);
}
";" {
    printf("%s : Semicolon\n", yytext);
}
"," {
    printf("%s : Comma\n", yytext);
}
"("|")" {
    printf("%s : Parentheses\n", yytext);
}
"["|"]" {
    printf("%s : Square bracket\n", yytext);
}
"{"|"}" {
    printf("%s : Curly brace\n", yytext);
}
%%
int main()
{
        FILE *fp;
        char file[30];
        printf("\nEnter Filename: ");
        scanf("%s", file);
        fp = fopen(file, "r");
        yyin = fp;
        yylex();
        printf("\nAnd there are %d whitespaces.\n\n",q);
        return 0;
}
```

OUTPUT

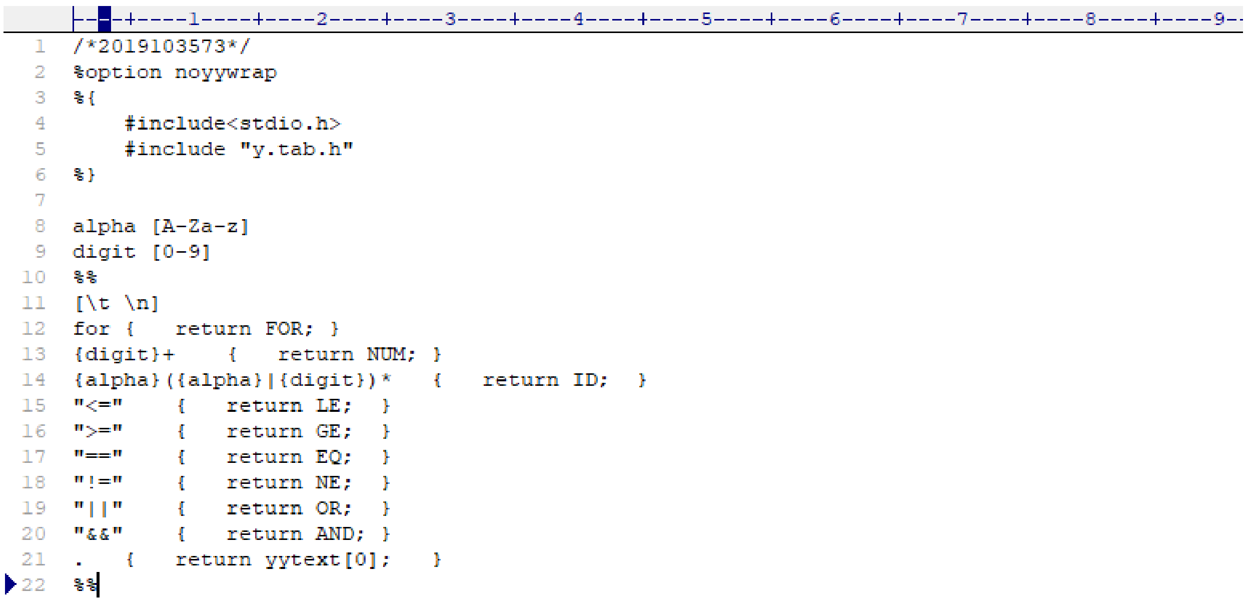


```
D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>lex T35732a.l

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

D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>a.exe

Enter Filename: INPUT35732a.txt
int : Keyword
i : Identifier
, : Comma
j : Identifier
, : Comma
a : Identifier
[ : Square bracket
2 : Number
] : Square bracket
[ : Square bracket
3 : Number
] : Square bracket
; : Semicolon
```


```
float : Keyword
c : Identifier
, : Comma
x : Identifier
; : Semicolon
for : Keyword
( : Parentheses
i : Identifier
= : Assignment operator
1 : Number
; : Semicolon
i : Identifier
<= : Relational operator
10 : Number
; : Semicolon
i : Identifier
++ : Post-increment Arithmetic operator
) : Parentheses
{ : Curly brace
for : Keyword
( : Parentheses
j : Identifier
= : Assignment operator
1 : Number
; : Semicolon
j : Identifier
<= : Relational operator
10 : Number
; : Semicolon
j : Identifier
++ : Post-increment Arithmetic operator
) : Parentheses
{ : Curly brace
a : Identifier
[ : Square bracket
i : Identifier
] : Square bracket
[ : Square bracket
j : Identifier
] : Square bracket
= : Assignment operator
1 : Number
; : Semicolon
x : Identifier
= : Assignment operator
c : Identifier
+ : Arithmetic operator
a : Identifier
[ : Square bracket
i : Identifier
] : Square bracket
[ : Square bracket
j : Identifier
] : Square bracket
; : Semicolon
} : Curly brace
} : Curly brace

And there are 48 whitespaces.
```

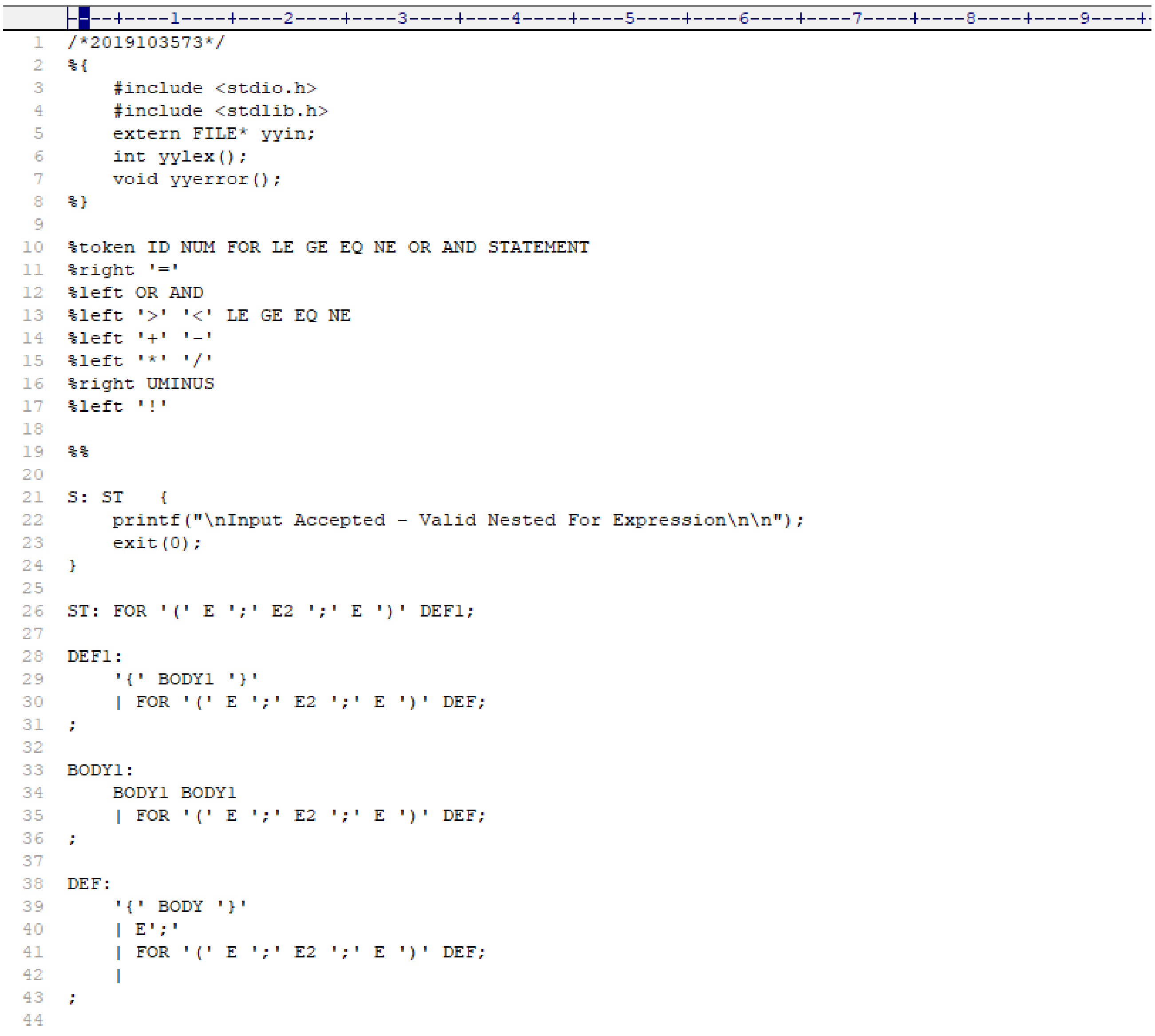
b. Validate the constructs in the program

LEX



```
/*2019103573*/
%option noyywrap
%{
    #include<stdio.h>
    #include "y.tab.h"
%}

alpha [A-Za-z]
digit [0-9]
%%
[\t \n]
for {   return FOR; }
{digit}+     {    return NUM; }
{alpha}({alpha}|{digit})*    {    return ID;   }
"<="     {   return LE;   }
">="     {   return GE;   }
"=="     {   return EQ;   }
"!="     {   return NE;   }
"||"     {   return OR;   }
"&&"     {   return AND;  }
.    {   return yytext[0];    }
%%
```

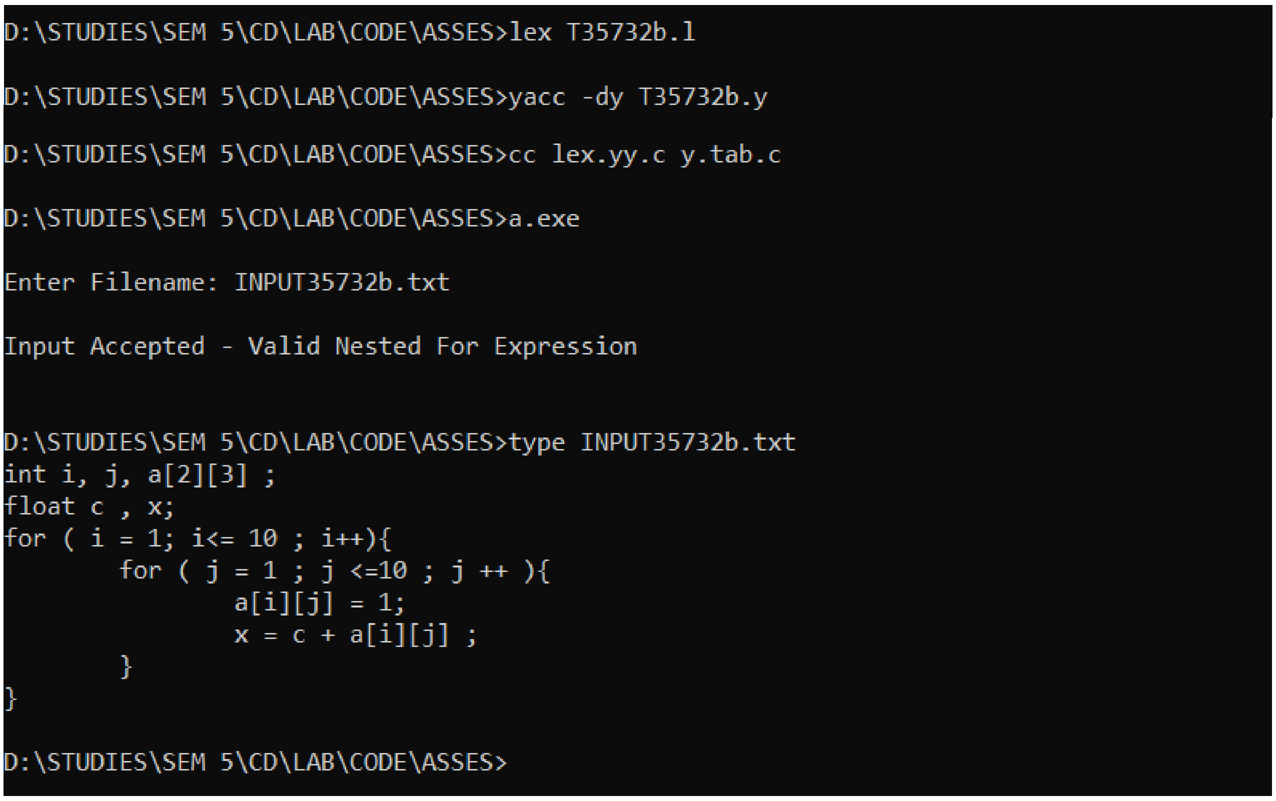
YACC



```
/*2019103573*/
%{
    #include <stdio.h>
    #include <stdlib.h>
    extern FILE* yyin;
    int yylex();
    void yyerror();
%}

%token ID NUM FOR LE GE EQ NE OR AND STATEMENT
%right '='
%left OR AND
%left '>' '<' LE GE EQ NE
%left '+' '-'
%left '*' '/'
%right UMINUS
%left '!'

%%

S: ST   {
    printf("\nInput Accepted - Valid Nested For Expression\n\n");
    exit(0);
}

ST: FOR '(' E ';' E2 ';' E ')' DEF1;

DEF1:
    '{' BODY1 '}'
    | FOR '(' E ';' E2 ';' E ')' DEF;
;

BODY1:
    BODY1 BODY1
    | FOR '(' E ';' E2 ';' E ')' DEF;
;

DEF:
    '{' BODY '}'
    | E';'
    | FOR '(' E ';' E2 ';' E ')' DEF;
    |
;
```

```
BODY:
    BODY BODY
    | E ';'
    | FOR '(' E ';' E2 ';' E ')' DEF;
    |
;

E:
    ID '=' E
    | E '+' E
    | E '-' E
    | E '*' E
    | E '/' E
    | E '<' E
    | E '>' E
    | E LE E
    | E GE E
    | E EQ E
    | E NE E
    | E OR E
    | E AND E
    | E '+' '+'
    | E '-' '-'
    | ID
    | NUM
;

E2:
    E'<'E
    | E'>'E
    | E LE E
    | E GE E
    | E EQ E
    | E NE E
    | E OR E
    | E AND E
;

%%

int main() {
    //FILE *fp;
        char file[30];
        printf("\nEnter Filename: ");
        scanf("%s", file);
        yyin = fopen(file, "r");
        //yylex();
    yyparse();
    return 0;
}

void yyerror() {
    printf("\nInvalid syntax - Invalid nested for expression\n\n");
}
```

OUTPUT



```
D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>lex T35732b.l

D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>yacc -dy T35732b.y

D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>cc lex.yy.c y.tab.c

D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>a.exe

Enter Filename: INPUT35732b.txt

Input Accepted - Valid Nested For Expression


D:\STUDIES\SEM 5\CD\LAB\CODE\ASSES>type INPUT35732b.txt
int i, j, a[2][3] ;
float c , x;
for ( i = 1; i<= 10 ; i++){
        for ( j = 1 ; j <=10 ; j ++ ){
                a[i][j] = 1;
                x = c + a[i][j] ;
        }
}

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