CS242-Assignment 3

Vatsal Gupta
200101105

Sweeya Reddy
200101079

README

8th Matrch 2023

ZIP FILE CONTENTS

- README.pdf(here)
- A3_40.l
- A3_40.nc
- A3_40.c
- Makefile
- Assignment3.pdf
- Output.txt ***(created)***
- A3_40.o ***(created)***
- a.out ***(created)***
- lex.yy.c ***(created)***

PROGRAMMING LANGUAGE USED:

nanoC, C, lex, (parsing)Bison.

Key Points/Pre-requisites

- Please make sure the terminal is run in the same directory (cwd) as the two files
- The program will create a .c file lex.yy.c from A3_40.l, lex file.

Commands to enter and expected output:

```
make output.txt (or) make
```

- Simply type make or make output.txt into the command line and a file named output.txt is created along with other supporter files.
- You can also retest the output by running make clean first which removes the optional files and then run make again which recreates them proving the sample output is indeed correct.

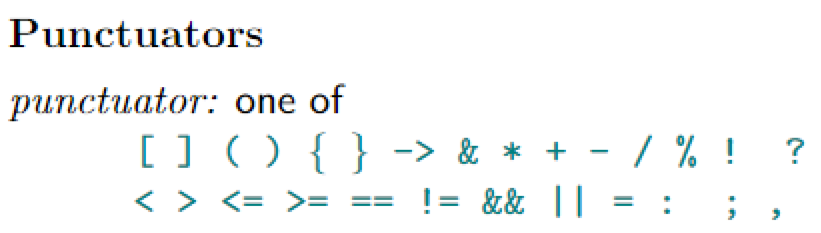
```
make clean
```

- Note that the make file works by essentially first using flex to create lex.yy.c file
- Then, this is used in the tester A3_40.c file to combine the two and create an a.out file
- To this a.out file, A3_40.nc file is fed and the lexer output is printed in the output.txt file in the form

```
<token-class,token-id,token-string which matched(yytext)>
```

Note:

- The lexer follows a certain priority order as given in the assignment
- The string literals to be terminated by \0 will be done at parser level and for now String literals are foud solely in the c-type form of simply specifying a string within double quotes
- Escape sequences are interpreted as the matching of instances like \? Instead of just ?
- **BONUS:** Some extra punctuations are handled and you may see ids like <?,token-id,?>
- FOr the punctuator class, we use separate token although this may be changed quite easily by changing <QUESTION,,> to <PUNCTUATOR : QUESTION,,>. We have followed a different convention though the lexer still matches the expression as a pucntuator due to the following regex, which was explicitly defined.



Punctuators

punctuator: one of

```
[ ] ( ) { } -> & * + - / % ! ?
< > <= >= == != && || = : ; ,
```

```
PUNCTUATOR
"["|"]"|"("|")"|"{"|"}"|"->"|"&"|"*"|"+"|"-"|"/"|"%"|"!"|"?"|"<"|">"|"<="|
">="|"=="|"!="|"&&"|"||"|"="|":"|";"|","
```

- If used incorrectly, the make file may change a source file like A3_40.c make sure that you use the makefile appropriately, if not then use the code provided below again to re-fill the A3_40.c file.

Files:

A3_40.c

```
#include <stdio.h>
#include "lex.yy.c"
extern char* yytext;
#define pr(x) printf(x, token, yytext)
int yywrap(){}

int main()
{
  int token;
  while((token=yylex()))
  {

         if ( token == SINGLE_LINE_COMM)    pr("< SINGLE_LINE_COMM, %d, %s>\n");
         else if ( token == MULTI_LINE_COMM)     pr("< MULTI_LINE_COMM, %d, %s>\n");

        //KeyWords
         else if ( token == RETURN) pr("< KEYWORD: RETURN, %d, %s >\n");
         else if ( token == VOID) pr("< KEYWORD: VOID, %d, %s >\n");
         else if ( token == CHAR) pr("< KEYWORD: CHAR, %d, %s >\n");
         else if ( token == FOR) pr("< KEYWORD: FOR, %d, %s >\n");
         else if ( token == IF) pr("< KEYWORD: IF, %d, %s >\n");
         else if ( token == INT) pr("< KEYWORD: INT, %d, %s >\n");
         else if ( token == ELSE) pr("< KEYWORD: ELSE, %d, %s >\n");

        // identifiers
         else if ( token == IDENTIFIER)     pr("< IDENTIFIER, %d, %s>\n");
         else if ( token == INTEGER_CONSTANT)       pr("< INTEGER_CONSTANT, %d, %s>\n");
         else if ( token == CHARACTER_CONSTANT)    pr("< CHARACTER_CONSTANT, %d, %s>\n");
         else if ( token == STRING_LITERAL)    pr("< STRING_LITERAL, %d, %s>\n");

        //punctuators
         else if ( token == SQRBROPEN)      pr("< SQRBROPEN, %d, %s>\n");
         else if ( token == SQRBRCLOSE)     pr("< SQRBRCLOSE, %d, %s>\n");
         else if ( token == RORBROPEN)      pr("< RORBROPEN, %d, %s>\n");
         else if ( token == RORBRCLOSE)     pr("< RORBRCLOSE, %d, %s>\n");
         else if ( token == CURBROPEN)     pr("< CURBROPEN, %d, %s>\n");
         else if ( token == CURBRCLOSE)    pr("< CURBRCLOSE, %d, %s>\n");
         else if ( token == DOT)    pr("< DOT, %d, %s>\n");
         else if ( token == ARWCOM)    pr("< ARWCOM, %d, %s>\n");
         else if ( token == AMPSND)    pr("< AMPSND, %d, %s>\n");
         else if ( token == MUL)    pr("< MUL, %d, %s>\n");
         else if ( token == ADD)    pr("< ADD, %d, %s>\n");
         else if ( token == SUB)    pr("< SUB, %d, %s>\n");
         else if ( token == NEG)    pr("< NEG, %d, %s>\n");
         else if ( token == EXCLAIM)    pr("< EXCLAIM, %d, %s>\n");
         else if ( token == DIV)    pr("< DIV, %d, %s>\n");
         else if ( token == MODULO)     pr("< MODULO, %d, %s>\n");
         else if ( token == LST)     pr("< LST, %d, %s>\n");
         else if ( token == GRT)     pr("< GRT, %d, %s>\n");
         else if ( token == LTE)    pr("< LTE, %d, %s>\n");
         else if ( token == GTE)    pr("< GTE, %d, %s>\n");
         else if ( token == EQL)     pr("< EQL, %d, %s>\n");
         else if ( token == NEQ)    pr("< NEQ, %d, %s>\n");
         else if ( token == AND)    pr("< AND, %d, %s>\n");
         else if ( token == OR)     pr("< OR, %d, %s>\n");
         else if ( token == QUESTION)       pr("< QUESTION, %d, %s>\n");
         else if ( token == COLON)      pr("< COLON, %d, %s>\n");
         else if ( token == SEMICOLON)      pr("< SEMICOLON, %d, %s>\n");
         else if ( token == ASSIGN)     pr("< ASSIGN, %d, %s>\n");
         else if ( token == STAREQ)     pr("< STAREQ, %d, %s>\n");
         else if ( token == DIVEQ)      pr("< DIVEQ, %d, %s>\n");
         else if ( token == MODEQ)      pr("< MODEQ, %d, %s>\n");
         else if ( token == PLUSEQ)     pr("< PLUSEQ, %d, %s>\n");
         else if ( token == MINUSEQ)    pr("< MINUSEQ, %d, %s>\n");
         else if ( token == COMMA)      pr("< COMMA, %d, %s>\n");


  }
return 0;
}
```

A3_40.l (lexer) regex expressions along with the complete file is attached

A3_40.nc (Nano C type file created for sample output generation and subject to lexical analysis.)

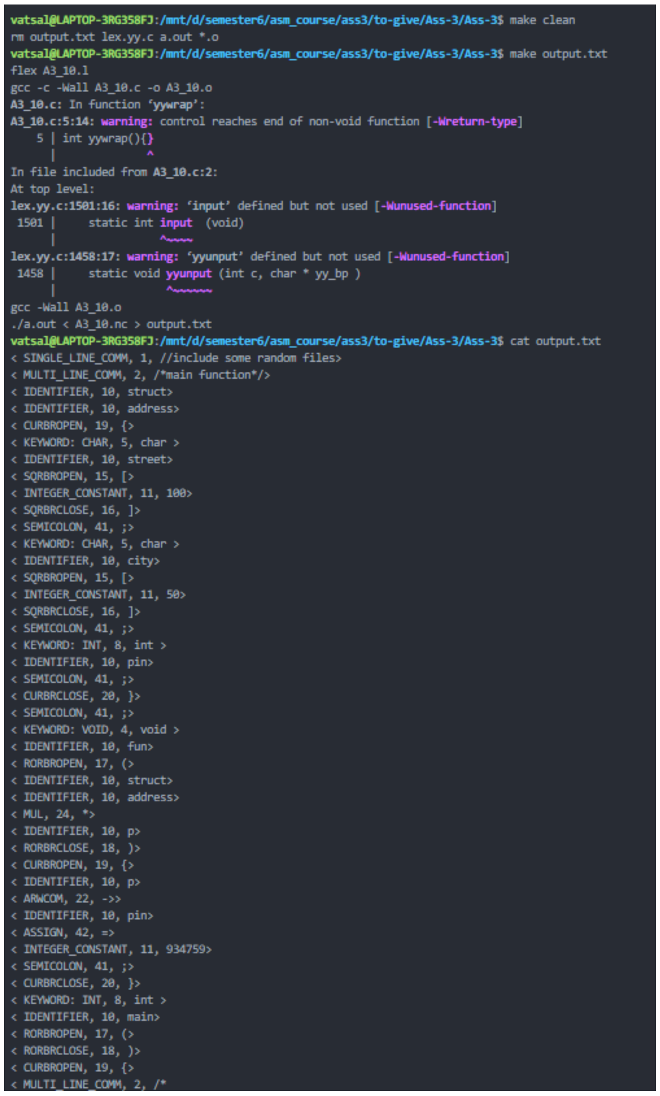
```
//include some random files
/*main function*/
struct address
{
   char street[100];
   char city[50];
   int pin;
};

void fun(struct address* p){
    p->pin=934759;
}

int main()
{
    /*
    Group 10 is amazing
    Sweeya and Vatsal have worked hard to develop this project. Kindly go through the README.
    */
    int x=0, z=5;
    char y = '0';
    if(x!=y){
        printf("x is not equal to y");
    }
    else{
        printf("x is equal to y");
    }

    z=(x+z*2)^2;
    x=x-z;
    z=z&&x;
    x=~x;
    z=z>>2;
    x=x<<2;
    z=z/2;
    x=x%2;
    z=z||0;
    if(x>=z){
        printf("x is greater than or equal to z");
    }
    if(x<=z){
        printf("z is greater than or equal to x");
    }
    if(x==z){
        printf("z is equal to x");
    }
    int *q=&z;
    int random_num=10;
    random_num+=1;
    random_num*=4.5;
    random_num-=25;
    random_num/=2;
    random_num%=7;
    int arr[1];
    for(int i=0; i<1; i++){
        arr[0]=0;
    }
    !(x == 0)?(random_num=0):(random_num=1);
    return 0;
}
```

Full Output as in output.txt

```
< SINGLE_LINE_COMM, 1, //include some random files>
< MULTI_LINE_COMM, 2, /*main function*/>
< IDENTIFIER, 10, struct>
< IDENTIFIER, 10, address>
< CURBROPEN, 19, {>
< KEYWORD: CHAR, 5, char >
< IDENTIFIER, 10, street>
< SQRBROPEN, 15, [>
< INTEGER_CONSTANT, 11, 100>
< SQRBRCLOSE, 16, ]>
< SEMICOLON, 41, ;>
< KEYWORD: CHAR, 5, char >
< IDENTIFIER, 10, city>
< SQRBROPEN, 15, [>
< INTEGER_CONSTANT, 11, 50>
< SQRBRCLOSE, 16, ]>
< SEMICOLON, 41, ;>
< KEYWORD: INT, 8, int >
< IDENTIFIER, 10, pin>
< SEMICOLON, 41, ;>
< CURBRCLOSE, 20, }>
< SEMICOLON, 41, ;>
< KEYWORD: VOID, 4, void >
< IDENTIFIER, 10, fun>
< RORBROPEN, 17, (>
< IDENTIFIER, 10, struct>
< IDENTIFIER, 10, address>
< MUL, 24, *>
< IDENTIFIER, 10, p>
< RORBRCLOSE, 18, )>
< CURBROPEN, 19, {>
< IDENTIFIER, 10, p>
< ARWCOM, 22, ->>
< IDENTIFIER, 10, pin>
< ASSIGN, 42, =>
< INTEGER_CONSTANT, 11, 934759>
< SEMICOLON, 41, ;>
< CURBRCLOSE, 20, }>
< KEYWORD: INT, 8, int >
< IDENTIFIER, 10, main>
< RORBROPEN, 17, (>
< RORBRCLOSE, 18, )>
< CURBROPEN, 19, {>
< MULTI_LINE_COMM, 2, /*
        Group 10 is amazing
        Sweeya and Vatsal have worked hard to develop this project. Kindly go
```

```
through the README.
       */>
< KEYWORD: INT, 8, int >
< IDENTIFIER, 10, x>
< ASSIGN, 42, =>
< INTEGER_CONSTANT, 11, 0>
< COMMA, 48, ,>
< IDENTIFIER, 10, z>
< ASSIGN, 42, =>
< INTEGER_CONSTANT, 11, 5>
< SEMICOLON, 41, ;>
< KEYWORD: CHAR, 5, char >
< IDENTIFIER, 10, y>
< ASSIGN, 42, =>
< CHARACTER_CONSTANT, 12, '0'>
< SEMICOLON, 41, ;>
< KEYWORD: IF, 7, if >
< RORBROPEN, 17, (>
< IDENTIFIER, 10, x>
< NEQ, 36, !=>
< IDENTIFIER, 10, y>
< RORBRCLOSE, 18, )>
< CURBROPEN, 19, {>
< IDENTIFIER, 10, printf>
< RORBROPEN, 17, (>
< STRING_LITERAL, 13, "x is not equal to y">
< RORBRCLOSE, 18, )>
< SEMICOLON, 41, ;>
< CURBRCLOSE, 20, }>
< KEYWORD: ELSE, 9, else >
< CURBROPEN, 19, {>
< IDENTIFIER, 10, printf>
< RORBROPEN, 17, (>
< STRING_LITERAL, 13, "x is equal to y">
< RORBRCLOSE, 18, )>
< SEMICOLON, 41, ;>
< CURBRCLOSE, 20, }>
< IDENTIFIER, 10, z>
< ASSIGN, 42, =>
< RORBROPEN, 17, (>
< IDENTIFIER, 10, x>
< ADD, 25, +>
< IDENTIFIER, 10, z>
< MUL, 24, *>
< INTEGER_CONSTANT, 11, 2>
< RORBRCLOSE, 18, )>
^< INTEGER_CONSTANT, 11, 2>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, x>
< ASSIGN, 42, =>
< IDENTIFIER, 10, x>
< SUB, 26, ->
< IDENTIFIER, 10, z>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, z>
< ASSIGN, 42, =>
< IDENTIFIER, 10, z>
< AND, 37, &&>
< IDENTIFIER, 10, x>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, x>
< ASSIGN, 42, =>
< NEG, 27, ~>
< IDENTIFIER, 10, x>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, z>
< ASSIGN, 42, =>
< IDENTIFIER, 10, z>
< GRT, 32, >>
< GRT, 32, >>
< INTEGER_CONSTANT, 11, 2>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, x>
< ASSIGN, 42, =>
< IDENTIFIER, 10, x>
< LST, 31, <>
< LST, 31, <>
< INTEGER_CONSTANT, 11, 2>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, z>
< ASSIGN, 42, =>
< IDENTIFIER, 10, z>
< DIV, 29, />
< INTEGER_CONSTANT, 11, 2>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, x>
< ASSIGN, 42, =>
< IDENTIFIER, 10, x>
< MODULO, 30, %>
< INTEGER_CONSTANT, 11, 2>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, z>
< ASSIGN, 42, =>
< IDENTIFIER, 10, z>
< OR, 38, ||>
< INTEGER_CONSTANT, 11, 0>
< SEMICOLON, 41, ;>
< KEYWORD: IF, 7, if >
< RORBROPEN, 17, (>
< IDENTIFIER, 10, x>
< GTE, 34, >=>
< IDENTIFIER, 10, z>
< RORBRCLOSE, 18, )>
< CURBROPEN, 19, {>
< IDENTIFIER, 10, printf>
< RORBROPEN, 17, (>
< STRING_LITERAL, 13, "x is greater than or equal to z">
< RORBRCLOSE, 18, )>
< SEMICOLON, 41, ;>
< CURBRCLOSE, 20, }>
< KEYWORD: IF, 7, if >
< RORBROPEN, 17, (>
< IDENTIFIER, 10, x>
< LTE, 33, <=>
< IDENTIFIER, 10, z>
< RORBRCLOSE, 18, )>
< CURBROPEN, 19, {>
< IDENTIFIER, 10, printf>
< RORBROPEN, 17, (>
< STRING_LITERAL, 13, "z is greater than or equal to x">
< RORBRCLOSE, 18, )>
< SEMICOLON, 41, ;>
< CURBRCLOSE, 20, }>
< KEYWORD: IF, 7, if >
< RORBROPEN, 17, (>
< IDENTIFIER, 10, x>
< EQL, 35, ==>
< IDENTIFIER, 10, z>
< RORBRCLOSE, 18, )>
< CURBROPEN, 19, {>
< IDENTIFIER, 10, printf>
< RORBROPEN, 17, (>
< STRING_LITERAL, 13, "z is equal to x">
< RORBRCLOSE, 18, )>
< SEMICOLON, 41, ;>
< CURBRCLOSE, 20, }>
< KEYWORD: INT, 8, int >
< MUL, 24, *>
< IDENTIFIER, 10, q>
< ASSIGN, 42, =>
< AMPSND, 23, &>
< IDENTIFIER, 10, z>
< SEMICOLON, 41, ;>
< KEYWORD: INT, 8, int >
< IDENTIFIER, 10, random_num>
< ASSIGN, 42, =>
< INTEGER_CONSTANT, 11, 10>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, random_num>
< PLUSEQ, 46, +=>
< INTEGER_CONSTANT, 11, 1>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, random_num>
< STAREQ, 43, *=>
< INTEGER_CONSTANT, 11, 4>
< DOT, 21, .>
< INTEGER_CONSTANT, 11, 5>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, random_num>
< MINUSEQ, 47, -=>
< INTEGER_CONSTANT, 11, 25>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, random_num>
< DIVEQ, 44, /=>
< INTEGER_CONSTANT, 11, 2>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, random_num>
< MODEQ, 45, %=>
< INTEGER_CONSTANT, 11, 7>
< SEMICOLON, 41, ;>
< KEYWORD: INT, 8, int >
< IDENTIFIER, 10, arr>
< SQRBROPEN, 15, [>
< INTEGER_CONSTANT, 11, 1>
< SQRBRCLOSE, 16, ]>
< SEMICOLON, 41, ;>
< KEYWORD: FOR, 6, for >
< RORBROPEN, 17, (>
< KEYWORD: INT, 8, int >
< IDENTIFIER, 10, i>
< ASSIGN, 42, =>
< INTEGER_CONSTANT, 11, 0>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, i>
< LST, 31, <>
< INTEGER_CONSTANT, 11, 1>
< SEMICOLON, 41, ;>
< IDENTIFIER, 10, i>
< ADD, 25, +>
< ADD, 25, +>
< RORBRCLOSE, 18, )>
< CURBROPEN, 19, {>
< IDENTIFIER, 10, arr>
< SQRBROPEN, 15, [>
< INTEGER_CONSTANT, 11, 0>
< SQRBRCLOSE, 16, ]>
< ASSIGN, 42, =>
< INTEGER_CONSTANT, 11, 0>
< SEMICOLON, 41, ;>
< CURBRCLOSE, 20, }>
< EXCLAIM, 28, !>
< RORBROPEN, 17, (>
< IDENTIFIER, 10, x>
< EQL, 35, ==>
< INTEGER_CONSTANT, 11, 0>
< RORBRCLOSE, 18, )>
< QUESTION, 39, ?>
< RORBROPEN, 17, (>
< IDENTIFIER, 10, random_num>
< ASSIGN, 42, =>
< INTEGER_CONSTANT, 11, 0>
< RORBRCLOSE, 18, )>
< COLON, 40, :>
< RORBROPEN, 17, (>
< IDENTIFIER, 10, random_num>
< ASSIGN, 42, =>
< INTEGER_CONSTANT, 11, 1>
< RORBRCLOSE, 18, )>
< SEMICOLON, 41, ;>
< KEYWORD: RETURN, 3, return >
< INTEGER_CONSTANT, 11, 0>
< SEMICOLON, 41, ;>
< CURBRCLOSE, 20, }>
```

Output Screenshot:



```
vatsal@LAPTOP-3RG358FJ:/mnt/d/semester6/asm_course/ass3/to-give/Ass-3/Ass-3$ make clean
rm output.txt lex.yy.c a.out *.o
vatsal@LAPTOP-3RG358FJ:/mnt/d/semester6/asm_course/ass3/to-give/Ass-3/Ass-3$ make output.txt
flex A3_10.l
gcc -c -Wall A3_10.c -o A3_10.o
A3_10.c: In function ‘yywrap’:
A3_10.c:5:14: warning: control reaches end of non-void function [-Wreturn-type]
    5 | int yywrap(){}
      |              ^
In file included from A3_10.c:2:
At top level:
lex.yy.c:1501:16: warning: ‘input’ defined but not used [-Wunused-function]
 1501 |     static int input  (void)
      |                ^~~~~
lex.yy.c:1458:17: warning: ‘yyunput’ defined but not used [-Wunused-function]
 1458 |     static void yyunput (int c, char * yy_bp )
      |                 ^~~~~~~
gcc -Wall A3_10.o
./a.out < A3_10.nc > output.txt
vatsal@LAPTOP-3RG358FJ:/mnt/d/semester6/asm_course/ass3/to-give/Ass-3/Ass-3$ cat output.txt
< SINGLE_LINE_COMM, 1, //include some random files>
< MULTI_LINE_COMM, 2, /*main function*/>
< IDENTIFIER, 10, struct>
< IDENTIFIER, 10, address>
< CURBROPEN, 19, {>
< KEYWORD: CHAR, 5, char >
< IDENTIFIER, 10, street>
< SQRBROPEN, 15, [>
< INTEGER_CONSTANT, 11, 100>
< SQRBRCLOSE, 16, ]>
< SEMICOLON, 41, ;>
< KEYWORD: CHAR, 5, char >
< IDENTIFIER, 10, city>
< SQRBROPEN, 15, [>
< INTEGER_CONSTANT, 11, 50>
< SQRBRCLOSE, 16, ]>
< SEMICOLON, 41, ;>
< KEYWORD: INT, 8, int >
< IDENTIFIER, 10, pin>
< SEMICOLON, 41, ;>
< CURBRCLOSE, 20, }>
< SEMICOLON, 41, ;>
< KEYWORD: VOID, 4, void >
< IDENTIFIER, 10, fun>
< RORBROPEN, 17, (>
< IDENTIFIER, 10, struct>
< IDENTIFIER, 10, address>
< MUL, 24, *>
< IDENTIFIER, 10, p>
< RORBRCLOSE, 18, )>
< CURBROPEN, 19, {>
< IDENTIFIER, 10, p>
< ARWCOM, 22, ->>
< IDENTIFIER, 10, pin>
< ASSIGN, 42, =>
< INTEGER_CONSTANT, 11, 934759>
< SEMICOLON, 41, ;>
< CURBRCLOSE, 20, }>
< KEYWORD: INT, 8, int >
< IDENTIFIER, 10, main>
< RORBROPEN, 17, (>
< RORBRCLOSE, 18, )>
< CURBROPEN, 19, {>
< MULTI_LINE_COMM, 2, /*
```

THANKS.