CECS 342 - Lab assignment 1 - Lexical Analyzer

Due date: Wednesday, February 14

Team Members: Bryan Tineo & Maxwell Guillermo

Completion of Lab Assignment:
Both team members contributed equally and collaborated throughout the completion of the lab assignment.

Code lab1.c

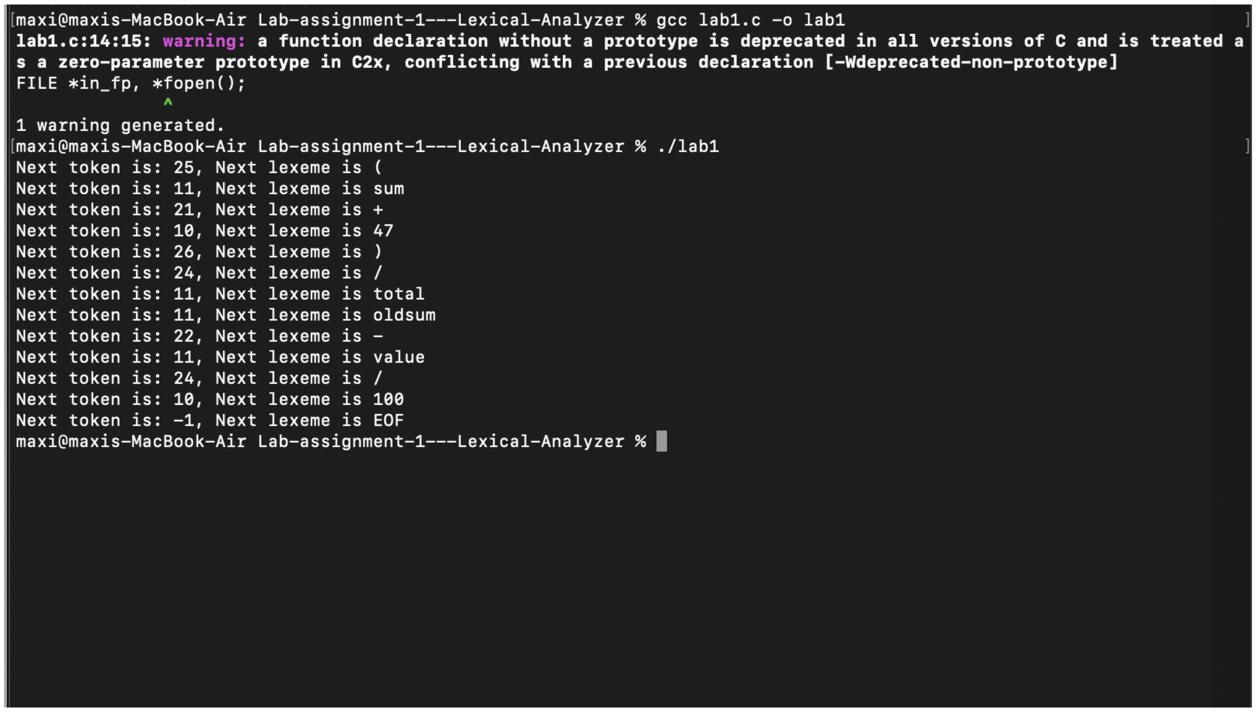
```c
/* A lexical analyzer system for simple
arithmetic expressions */
#include <stdio.h>
#include <ctype.h>
/* Global declarations */
/* Variables */
int charClass;
//  if given this:  (sum + 47) / total
char lexeme[100]; // <-- [(,s,u,m,+,4,7,),/,t,o,t,a,l] This array is actually a whole
string. According to c/c++ this will be treated as both a whole null-terminated string
and a list of characters
char nextChar;
int lexLen; //<- initialize lexlen to keep treak of the length of the whole input from
txt
int token;
int nextToken;
FILE *in_fp, *fopen();
/* Function declarations */
void addChar();
void getChar();
void getNonBlank();
int lex();
/* Character classes */
#define LETTER 0
#define DIGIT 1
#define UNKNOWN 99
/* Token codes */
```

```c
#define INT_LIT 10
#define IDENT 11
#define ASSIGN_OP 20
#define ADD_OP 21
#define SUB_OP 22
#define MULT_OP 23
#define DIV_OP 24
#define LEFT_PAREN 25
#define RIGHT_PAREN 26
/******************************************************/
/* main driver */
int main()
{
     /* Open the input data file and process its contents */
     if ((in_fp = fopen("front.txt", "r")) == NULL)
           printf("ERROR - cannot open front.in \n");
     else
     {
           getChar();
           do
           {
                 lex();
           } while (nextToken != EOF);
     }
     return 0;
}
/*****************************************************/
/* lookup - a function to lookup operators and parentheses
and return the token */
int lookup(char ch)
{
     switch (ch)
     {
     case '(':
           addChar();
           nextToken = LEFT_PAREN;
           break;
     case ')':
           addChar();
           nextToken = RIGHT_PAREN;
           break;
     case '*':
           addChar();
           nextToken = MULT_OP;
           break;
     case '/':
           addChar();
           nextToken = DIV_OP;
           break;
     case '+':
           addChar();
           nextToken = ADD_OP;
           break;
     case '-':
           addChar();
           nextToken = SUB_OP;
           break;
     case '=':
           addChar();
           nextToken = ASSIGN_OP;
           break;
     }

     return nextToken;
}

/*****************************************************/
/* addChar - a function to add nextChar to lexeme */
void addChar()
{
     if (lexLen <= 98)
     {
           // At the start of the program
           // lexLen = 0 that was assigned on lex function
           // when we are her
           // We are actually doing this:
           lexeme[lexLen++] = nextChar; // ex) when lexLen = 0  than, lexeme[0] = "("
           // After assigning "(" into the zero position the ++ (post i-increment) on
lexLen++ will update the Lexlen
           // After assinging to 0 position the "lexLen" will become, in this case,
were we will store the next character
           // However, since c/c++ are known for null terminated string. Than we need
to create a null value after this added character
           lexeme[lexLen] = 0; // -> lexeme[1] =0
        // So at the first itration the lexeme array would look like this
        // lexeme[100] = ["(",0]
        // c/c++ treats this array as an array of chracters
    }
    else
        printf("Error - lexeme is too long \n");
}
/*****************************************************/
/* getChar - a function to get the next character of
input and determine its character class */
void getChar()
{
    if ((nextChar = getc(in_fp)) != EOF)
    {
        if (isalpha(nextChar))
            charClass = LETTER;
        else if (isdigit(nextChar))
            charClass = DIGIT;

        else
            charClass = UNKNOWN;
    }
    else
        charClass = EOF;
}
/*****************************************************/
/* getNonBlank - a function to call getChar until it
returns a non-whitespace character */
void getNonBlank()
{
    // ignoring white characters
    while (isspace(nextChar))
        getChar();
}
/*
*****************************************************/
/* lex - a simple lexical analyzer for arithmetic
expressions */

int lex()
{
```

```
lexLen = 0;     // length of lexeme
getNonBlank(); //

switch (charClass)
{
/* Parse identifiers */
case LETTER:
      addChar();
      getChar();
      while (charClass == LETTER || charClass == DIGIT)
      {
            addChar();
            getChar();
      }
      nextToken = IDENT;
      break;
case DIGIT:

      addChar();
      getChar();
      while (charClass == DIGIT)
      {
            addChar();
            getChar();
      }
      nextToken = INT_LIT;
      break;

case UNKNOWN:
      lookup(nextChar);
      getChar();

      break;
/* EOF */
// handles the end-of-file character
case EOF:
      nextToken = EOF; // <-- this assigns -1 as token for EOF
      lexeme[0] = 'E';
      lexeme[1] = 'O';
      lexeme[2] = 'F';
      lexeme[3] = 0; // Null-terminate the string
      break;
```

```c
    } /* End of switch */
    printf("Next token is: %d, Next lexeme is %s\n", nextToken, lexeme);
    return nextToken;
} /* End of function lex */
```

Code Output:



```
maxi@maxis-MacBook-Air Lab-assignment-1---Lexical-Analyzer % gcc lab1.c -o lab1
lab1.c:14:15: warning: a function declaration without a prototype is deprecated in all versions of C and is treated a
s a zero-parameter prototype in C2x, conflicting with a previous declaration [-Wdeprecated-non-prototype]
FILE *in_fp, *fopen();
              ^
1 warning generated.
maxi@maxis-MacBook-Air Lab-assignment-1---Lexical-Analyzer % ./lab1
Next token is: 25, Next lexeme is (
Next token is: 11, Next lexeme is sum
Next token is: 21, Next lexeme is +
Next token is: 10, Next lexeme is 47
Next token is: 26, Next lexeme is )
Next token is: 24, Next lexeme is /
Next token is: 11, Next lexeme is total
Next token is: 11, Next lexeme is oldsum
Next token is: 22, Next lexeme is -
Next token is: 11, Next lexeme is value
Next token is: 24, Next lexeme is /
Next token is: 10, Next lexeme is 100
Next token is: -1, Next lexeme is EOF
maxi@maxis-MacBook-Air Lab-assignment-1---Lexical-Analyzer %
```