Sonoma State University
Computer Science Department
CS 460 – Fall 2017 – Watts

Semester Project – Part 1 (aka Project 1)

For this part of the project you will be writing a lexical analyzer. Your program should be written in C++ using the Object Oriented Paradigm.

Specifications

1. Your lexical analyzer should recognize lexemes for identifiers, numeric literals and the symbols listed in the table on the second page of this document.
2. Symbols must start with a letter and may be followed by any number of letters, digits, or underscores.
3. Before writing any code, you should design a regular expression that describes the set of valid lexemes and a DFA that recognizes valid lexemes.
4. Your lexical analyzer should be table driven. The table can be hard coded or stored in an external file.
5. Your analyzer will read a source file (.ss) and produce 3 files:
   a listing (.lst) file
   a token (.p1) file
   a debugging(.dbg) file
   The listing file should contain the lines of the input file with line numbers and errors. The token file should contain a list of the tokens identified in the file ad their corresponding lexemes.
   The debugging file is for your use. I suggest that it also contain the lines of the input file with line numbers and errors AND, following each line, a list of the tokens found on the line and their corresponding lexemes. Sample input and output files are shown on the last page of this document.
6. Your analyzer should provide a header file containing an enumerated type called token and prototypes for functions available to other parts of your project:

```
/* The GetToken function returns the member of the enumerated
typeassociated with the next lexeme in the source file. */
token_type GetToken();

/* The GetLexeme function returns the most recently recognized
lexemefrom the source file. GetLexeme can only be called after
GetToken has been called at least once. GetLexeme cannot be
called after the end of the source file has been detected. */
string GetLexeme();

/* The GetTokenName function returns a string containing the
name of the token passed to it. */
string GeTokenName(token_type token);

/* The ReportError function will be used by the lexical
analyzer andother parts of your project to report an error. The
lexical analyzerwill print the error message following the line
of source code inwhich it was detected. */
void ReportError (const string & message);
```

7. A framework for this part of the project can be found in the folder Project1Framework in the course pickup folder.

Due date: Monday, 9 October 2017; 11:59 pm.
To turn in: A tarred and zipped directory containing the files needed to compile and execute your lexical analyzer. The directory should be called *Teama*P1 and the tarred and zipped file should be called *Teama*P1.tgz. Your directory should contain a makefile with a default target of P1.out.

Identifier	α(α|#|_)*	IDENT_T
Numeric Literals	(+|-|λ (#$^{+}$ | #*. #$^{+}$ | #$^{+}$.#*)	NUMLIT_T
Key words		
	cons	CONS_T
	if	IF_T
	cond	COND_T
	display	DISPLAY_T
	newline	NEWLINE_T
	and	AND_T
	or	OR_T
	not	NOT_T
	define	DEFINE_T
	(cad*r) | (cd^{+}r)	LISTOP_T
Predicates		
	number?	NUMBERP_T
	symbol?	SYMBOLP_T
	list?	LISTP_T
	zero?	ZEROP_T
	null?	NULLP_T
	char?	CHARP_T
	string?	STRINGP_T
Arithmetic		
	+ addition	PLUS_T
	- subtraction	MINUS_T
	/ division	DIV_T
	* multiplication	MULT_T
Logical/Relational		
	= Equal to	EQUALTO_T
	> Greater than	GT_T
	< Less than	LT_T
	>= Greater than or equal to	GTE_T
	<= Less than or equal to	LTE_T
Other		
	(Open Paren	LPAREN_T
	) Close Paren	RPAREN_T
	' Quote	QUOTE_T
		EOF_T

Sample Input file (P1-1.ss)

```
identifier ident_1 function
0 123 1.1 -1.1 +1.1 12. -.1

cons if cond display newline and or not define
car cdr cadr cddr ?number
number?  symbol?  list?$zero?  null?  char?  string?

+ - / *
= > < >= <=
( ) '
```

Sample Listing file (P1-1.lst)

```
Input file: P1-1.ss
   1: identifier ident_1 function
   2: 0 123 1.1 -1.1 +1.1 12. -.1
   3:
   4: cons if cond display newline and or not define
   5: car cdr cadr cddr ?number
Error at 5,19: Invalid character found: ?
   6: number?  symbol?  list?$zero?  null?  char?  string?
Error at 6,24: Invalid character found: $
   7:
   8: + - / *
   9: = > < >= <=
  10: ( ) '
2 errors found in input file
```

Sample Token file (P1-1.p1)

```
      IDENT_T         identifier              ERROR_T         ?
      IDENT_T         ident_1                 IDENT_T         number
      IDENT_T         function                NUMBERP_T       number?
      NUMLIT_T        0                       SYMBOLP_T       symbol?
      NUMLIT_T        123                     LISTP_T         list?
      NUMLIT_T        1.1                     ERROR_T         $
      NUMLIT_T        -1.1                    ZEROP_T         zero?
      NUMLIT_T        +1.1                    NULLP_T         null?
      NUMLIT_T        12.                     CHARP_T         char?
      NUMLIT_T        -.1                     STRINGP_T       string?
      CONS_T          cons                    PLUS_T          +
      IF_T            if                      MINUS_T         -
      COND_T          cond                    DIV_T           /
      DISPLAY_T       display                 MULT_T          *
      NEWLINE_T       newline                 EQUALTO_T       =
      AND_T           and                     GT_T            >
      OR_T            or                      LT_T            <
      NOT_T           not                     GTE_T           >=
      DEFINE_T        define                  LTE_T           <=
      LISTOP_T        car                     LPAREN_T        (
      LISTOP_T        cdr                     RPAREN_T        )
      LISTOP_T        cadr                    QUOTE_T         '
      LISTOP_T        cddr
```