Sonoma State University
Computer Science Department
CS 460 – Fall 2018– 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, 8 October 2018; 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 *Teamα*P1 and the tarred and zipped file should be called *Teamα*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
String Literals	" . . . "	STRLIT_T
Key words	(cad*r) | (cd^{+}r)	LISTOP_T
	cons	CONS_T
	if	IF_T
	cond	COND_T
	else	ELSE_T
	display	DISPLAY_T
	newline	NEWLINE_T
	and	AND_T
	or	OR_T
	not	NOT_T
	define	DEFINE_T
Predicates	number?	NUMBERP_T
	list?	LISTP_T
	zero?	ZEROP_T
	null?	NULLP_T
	string?	STRINGP_T
Arithmetic	+ addition	PLUS_T
	- subtraction	MINUS_T
	/ division	DIV_T
	* multiplication	MULT_T
	modulo	MODULO_T
	round	ROUND_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	SQUOTE_T
		ERROR_T
		EOF_T

Sample Input file (P1-0.ss)

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

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

+ - / * modulo round
= > < >= <=
( ) 'a
```

Sample Listing file (P1-0.lst)

```
Input file: P1-0.ss
   1: identifier ident_1 function
   2: 0 123 1.1 -1.1 +1.1 12. -.1
   3: "Hello World"
   4:
   5: cons if cond display newline and or not define
   6: car cdr cadr cddr ?number
Error at 6,19: Invalid character found: ?
   7: number?  list?$zero?  null?  string?
Error at 7,15: Invalid character found: $
   8:
   9: + - / * modulo round
  10: = > < >= <=
  11: ( ) 'a
  12:
2 errors found in input file
```

Sample Token file (P1-0.p1)

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