CS 3513 Programming Languages

Lab 1 : Implementing a Lexical Scanner for RPAL

This is a group project with 2 students in a group. Please use same group members as you selected in the Moodle group selection. All the submissions should have the .zip file as lab1_<index no 1>_<index no 2>.zip. Replace <index no x> with your index number.

1. Lab Objectives

1. Understand lexical analysis for functional languages
2. Recognize RPAL tokens
3. Implement DFA-based tokenization
4. Handle identifiers, integers, strings, operators, and keywords
5. Report lexical errors

2. RPAL Lexical Rules

A lexical scanner converts RPAL source code into tokens.

Example Input:

let x = 5 in x + 1

Output of the Tokens:

<KEYWORD, “let”>

<IDENTIFIER, “x”>

< OPERATOR, “=”>

<INTEGER, “5”>

<KEYWORD, “in”>

<IDENTIFIER, “x”>

<OPERATOR, “+”>

<INTEGER, “1”>

```
Identifier -> Letter (Letter | Digit | '_')*              => '<IDENTIFIER>';

Integer    -> Digit+                                     => '<INTEGER>';

Operator   -> Operator_symbol+                           => '<OPERATOR>';

String     -> ''''
                ( '\' 't' | '\' 'n' | '\' '\' | '\' ''''
                | '('     | ')'     | ';'     | ','
                | ' '
                | Letter | Digit | Operator_symbol
                )* ''''                                  => '<STRING>';

Spaces     -> ( ' ' | ht | Eol )+                        => '<DELETE>';

Comment    -> '//'
             ( '''' | '(' | ')' | ';' | ',' | '\' | ' '
                | ht | Letter | Digit | Operator_symbol
             )* Eol                                      => '<DELETE>';

Punction   -> '('                                        => '('
           -> ')'                                        => ')'
           -> ';'                                        => ';'
           -> ','                                        => ',';

Letter     -> 'A'..'Z' | 'a'..'z';

Digit      -> '0'..'9';

Operator_symbol -> '+' | '-' | '*' | '<' | '>' | '&' | '.'
           | '@' | '/' | ':' | '=' | '~' | '|' | '$'
           | '!' | '#' | '%' | 'ˆ' | '_' | '[' | ']'
           | '{' | '}' | '"' | '`' | '?';
```

Also use the below keywords list to identify whether an identifier is a keyword or not.

Keywords List:

Let, where, true, false, not, fn, ls, gr, ge, aug, le, nil, dummy, or, in, eq, ne, and, rec, within

3. Step-by-Step Guide

Step 1: Read Input File:
Read RPAL program from input file.
./a.out <filename>

Step 2: Define Token Structure:
Token should contain type and value.

Step 3: Define Keyword Set:
Store RPAL keywords in a list or set.

Step 4: Toekn Classification:
Functions to detect letters, digits, operators, keywords, integers, strings, operators and comments handling

Step 5: Main Loop:
Skip whitespace, then detect tokens.

Step 6: Error Reporting:
Report invalid symbols and unterminated strings.

4. Student Tasks

1. Implement scanner
2. Print tokens as <TYPE, VALUE>
3. Handle errors
4. Submit code (single .cpp file)

8. Marking Scheme (100 Marks)

Implementation (40): including Input handling, Identifier, Keywords, Integer, String, Operator, Comments

Output Correctness (50): 10 test cases and 5 marks for each.

Code Quality (10)

Example:

Input:

```
let Sum(A) = Psum (A,Order A )

                         where rec Psum (T,N) = N eq 0 -> 0

                             | Psum(T,N-1)+T N

in Print ( Sum (1,2,3,4,5) )
```

Output:

```
<KEYWORD, "let">
<IDENTIFIER, "Sum">
<PUNCTUATION, "(">
<IDENTIFIER, "Sum">
<PUNCTUATION, "(">
<IDENTIFIER, "A">
<PUNCTUATION, ")">
<OPERATOR, "=">
<IDENTIFIER, "Psum">
<PUNCTUATION, "(">
<IDENTIFIER, "A">
<PUNCTUATION, ",">
<IDENTIFIER, "Order">
<IDENTIFIER, "A">
<PUNCTUATION, ")">
<KEYWORD, "where">
<KEYWORD, "rec">
<IDENTIFIER, "Psum">
<PUNCTUATION, "(">
<IDENTIFIER, "Psum">
<PUNCTUATION, "(">
<IDENTIFIER, "T">
<PUNCTUATION, ",">
<IDENTIFIER, "N">
<PUNCTUATION, ")">
<OPERATOR, "=">
<IDENTIFIER, "N">
<KEYWORD, "eq">
<INTEGER, "0">
<OPERATOR, "->">
<INTEGER, "0">
<OPERATOR, "|">
<IDENTIFIER, "Psum">
<PUNCTUATION, "(">
<IDENTIFIER, "T">
<PUNCTUATION, ",">
<IDENTIFIER, "N">
<OPERATOR, "-">
<INTEGER, "1">
<PUNCTUATION, ")">
<OPERATOR, "+">
<IDENTIFIER, "T">
<IDENTIFIER, "N">
<KEYWORD, "in">
<IDENTIFIER, "Print">
<PUNCTUATION, "(">
<IDENTIFIER, "Sum">
<PUNCTUATION, "(">
<INTEGER, "1">
<PUNCTUATION, ",">
<INTEGER, "2">
<PUNCTUATION, ",">
<INTEGER, "3">
<PUNCTUATION, ",">
<INTEGER, "4">
<PUNCTUATION, ",">
<INTEGER, "5">
<PUNCTUATION, ")">
<PUNCTUATION, ")">
```