The following are instructions for project 1a

Write code for lexical analysis to recognize the following lexical items.

1.  The character set is:

ASN.1 characters A to Z (LATIN CAPITAL LETTER A to LATIN CAPITAL LETTER Z)

a to z (LATIN SMALL LETTER A to LATIN SMALL LETTER Z)

0 to 9 (DIGIT ZERO to DIGIT 9)

" (QUOTATION MARK)

( (LEFT PARENTHESIS)

) (RIGHT PARENTHESIS)

, (COMMA)

- (HYPHEN-MINUS)

: (COLON)

= (EQUALS SIGN)

{ (LEFT CURLY BRACKET)

| (VERTICAL LINE)

} (RIGHT CURLY BRACKET)

2. Reserved Words: TAGS, BEGIN, SEQUENCE, INTEGER, DATE, END

3. TOKENS: Range\_Seperator, ASSIGN, LCURLY, RCURLY, COMMA, LPAREN, RPAREN, TypeRef, Identifier,Number

1. A lexical item shall be separated from a following lexical item by one or more instances of white-space.
2. white-space - numbers represent the ASCII Code:

HORIZONTAL TABULATION (9)

LINE FEED (10)

VERTICAL TABULATION (11)

FORM FEED (12)

CARRIAGE RETURN (13)

SPACE (32)

1. Type references

Name of lexical item – typereference

A "typereference" shall consist of an arbitrary number (one or more) of letters, digits, and hyphens. **The initial character shall be an upper-case letter.** A hyphen shall not be the last character. A hyphen shall not be immediately followed by another hyphen.

NOTE – The rules concerning hyphen are designed to a void ambiguity with (possibly following) comment.

1. Identifiers

Name of lexical item – identifier

An "identifier" shall consist of an arbitrary number (one or more) of letters, digits, and hyphens. **The initial character shall be a lower-case letter**. A hyphen shall not be the last character. A hyphen shall not be immediately followed by another hyphen.

1. Numbers

Name of lexical item – number

A "number" shall consist of one or more digits. The first digit shall not be zero unless the "number" is a single digit.

1. Assignment lexical item

Name of lexical item – "::="

This lexical item shall consist of the sequence of characters: ::=

1. Range separator

Name of lexical item – ".."

This lexical item shall consist of the sequence of characters: ..

1. “{“ - LCURLY
2. “}” – RCURLY
3. “,” – COMMA
4. “(“ – LPAREN
5. “)” – RPAREN

The input String to test for success is:

MyShopPurchaseOrders TAGS ::= BEGIN  
  
PurchaseOrder ::= SEQUENCE {  
dateOfOrder DATE,  
customer    CustomerInfo,  
items       ListOfItems  
}  
  
  
Item ::= SEQUENCE {  
itemCode        INTEGER ( 1 .. 99999 ) ,  
power           INTEGER ( 110 | 220 ) ,  
deliveryTime    INTEGER ( 8 .. 12 | 14 .. 19 ) ,  
quantity        INTEGER ( 1 .. 1000 ) ,  
unitPrice       INTEGER ( 1 .. 9999 ) ,  
}  
END

Run it once on above code and it should be successful.

Run it a second time so that you make up some input string that will return some errors. For example insert some characters not part of the alphabet.

GRADING CRITERIA

1. Write the lexer in c/c++ or java only
2. Submit your code.
3. Submit a screen shot of your execution, showing the list of tokens and lexemes and SUCCESS
4. Submit a screen shot of your execution showing error.
5. You may use string library, But DO NOT use the regexp API of any such library.
6. A report on what you did for coding … did you make REGEX, DFA, or just code from REGEX… DO NOT USE FLEX/LEX