Concordia University
Department of Computer Science and Software Engineering

Compiler Design (COMP 442/6421)
Winter 2014

Assignment 1, Lexical Analyzer

Deadline:	Monday January 27th, 2014
Evaluation:	10% of final grade
Late submission:	penalty of 50% for each late working day

Design and implement a scanner for a programming language whose lexical specifications are given below. The scanner identifies and outputs tokens (valid words and punctuation) in the source program. Its output is a token that can thereafter be used by the syntactic analyzer to verify that the program is syntactically valid. When called, the lexical analyzer should extract the next token from the source program. The lexical analyzer should be able to output a token even if the input does not form a correct program. The syntax of the language will be specified later in assignment #2. Note that completeness of testing will be a major grading topic. You are responsible for providing appropriate test cases that test for a wide variety of valid and invalid cases.

Atomic lexical elements of the language

```
      id ::= letter alphanum*
     num ::= nonzero digit* fraction | 0 fraction
alphanum ::= letter | digit | _
fraction ::= .digit* nonzero | .0 | ε
  letter ::= a..z |A..Z
   digit ::= 0..9
 nonzero ::= 1..9
```

Operators, punctuation and reserved words

```
==      +       (       if
<>      -       )       then
<       *       {       else
>       /       }       for
<=      =       [       class
>=      and     ]       int
;       not     /*      float
,       or      */      get
.               //      put
                        return
```

Note : It is up to you to analyze this lexical definition and figure out if there are ambiguities or other errors. Any changes to this definition, if any, must be justified and should not result in diminishing the expressive power of the language. Also, keep in mind that you have to design the lexical analyzer in a flexible way, so that it can easily be adapted if changes are needed when designing the syntactic analyzer.

Note : The tokens // and /* followed */ by denote comments in the source code.

Work to be done

- Identify in your documentation the lexical conventions you will use in the design of the lexical analyzer especially the problems related to the presence of “white space” characters in the source code.
- Identify in your documentation the codes that you will use for tokens. Numbers or capital acronyms are normally used. Include the description of your convention in the documentation.
- Identify in your documentation all the possible lexical errors that the lexical analyzer might encounter. Identify the possible error recovery techniques you could use. Choose one of them and discuss in your documentation why you have chosen it.
- Write a lexical analyzer that recognizes the above tokens. It should be a function that returns a data structure containing the information about the next token identified in the source program file. The data structure should contain information such as (1) the token type (2) its value (or lexeme) when applicable and (3) its location in the source code. Fully describe this structure in your documentation.
- Include in your documentation a finite state machine describing the operation of your lexical analyzer.
- Include a driver that repeatedly calls the lexical analysis function and prints the token type of each token until the end of the source program is reached. The lexical analyzer should optionally print the token stream to a file (for verification purposes). Another file (also for verification purposes) should also contain representative error messages **each time** an error is encountered in the input program. The lexical analyzer should not stop after encountering an error. Error messages should be clear and should identify the location of the error in the source code.
- Include many test cases that test a wide variety of valid and invalid cases.

Assignment submission requirements and procedure

You have to submit your assignment before midnight on the due date using the ENCS Electronic Assignment Submission system under the category "*programming assignment 1*". The file submitted must be a **.zip** file containing:

- all your code
- a set of input files to be used for testing purpose, as well as a printout of the resulting output of the program for each input file
- a simple document containing the information requested above

You are also responsible to give proper compilation and execution instructions to the marker in a README file. If the marker cannot compile and execute your programs, you might have to have a meeting for a demonstration.

Evaluation criteria and grading scheme

Documentation:	
Description of lexical conventions	3 pt
Finite state automaton	5 pts
Program:	
Correct implementation according to assignment statement	15 pts
Output of clear error messages	5 pts
Output of token stream	2 pt
Error recovery	5 pts
Completeness of test cases	15 pts
Total	50 pts