**CSE 340 Principles of Programming Languages**

Spring 2014

**Programming Assignment 1**

Due Date: February 5th, 2014

**OBJECTIVE**

Create the lexical analyzer for a programming language.

**INSTRUCTIONS**

1. Write in C++ a code, which:
2. Receives two parameters: (1) the name of an input file and (2) the name of an output file.
3. Reads the input file line by line.
4. Divides each line in strings using whitespaces, delimiters, and operators.
5. Has the methods **isDelimiter(char)** and **isOperator(char)**. Those methods define the full set of delimiters and operators. They should return a boolean value.
6. Generates an output file, which name is the second parameter. The output file should have two columns; in the left column are the names of the identified tokens (see number 6 below for the whole list), and in the right column are the strings or words. The name of the tokens should be in UPPERCASES. The label “UNDEFINED” is used as a token for any string that does not match with any lexical rule. Columns are separated by a tab (\t) character.
7. Has, in a method called **lexer()**, the implementation of YOUR lexical analyzer. Lexical analyzer should be implemented using a DFA model. Details were provided in the lecture on Jan 27th. Make sure that your add proper comments to your code following the C++ or JavaDoc convention. Your lexical analyzer has to be able to recognize all following tokens: (1) OPERATOR, (2) DELIMITER, (3) INTEGER, (4) FLOAT, (5) HEXADECIMAL, (6) OCTAL, (7) STRING, (8) CHAR, and (9) ID. Also the following words must be identified as (10) KEYWORD: if, else, while, return, int, float, void, char, string, boolean.
8. Use the provided input file to test your lexical analyzer.
9. Read and actively participate in the discussion board about common errors in lexical analysis and use those discussions to improve your implementation.
10. Create a zip file, you should follow the following naming convention: **Firstname\_Lastname\_Projnumber.zip**, this file should contain the following:

* Your source code, the .cpp file. You should name the file as **Firstname\_Lastname\_Projnumber**
* The original input file provided and the output file for that. You should name the files as **Firstname\_Lastname\_Projnumber\_input.txt** and **Firstname\_Lastname\_Projnumber\_output.txt.**
* One new input file and its corresponding output file. You should name the files as **Firstname\_Lastname\_Projnumber\_input\_new.txt** and **Firstname\_Lastname\_Projnumber\_output\_new.txt.**
* A **makefile** to control the compilation of your code. The makefile should have at least a default target that builds your project. Google “makefile tutorial” for how to write makefiles. Before submitting your project, don’t forget to try the makefile at general.asu.edu and make sure it is fully working (able to compile and run).

Remember that:

* No credit will be given to programs that do not compile on general.
* No credit will be given to programs that do not execute correctly on general.
* No credit will be given if the output file does not follow the format described above.
* No credit will be given if any file (source file, makefile, or test cases) is missing.