Project 1: A Lexical Analyzer for TruPL

Write a direct-coded scanner that recognizes the tokens in TruPL. Your scanner will consist of header and implementation files that give definitions for the token types of TruPL, a buffer class that preprocesses the input to remove excess whitespace and comments, and a scanner class. The bulk of your scanner will be implemented within a function called next token(), a member function of the class Scanner. Next Token() returns a pointer to an object that is from a derived (sub) class of Token, as shown in the Scanner header file.

Each time next token() is called, it should return the next lexeme in the source file. When all the lexemes in the source file have been read, the next call to next token() should return an EofToken.

You should complete your project in the following phases:

1. Complete the class definitions for the derived (sub) classes of the Token class: KeywordToken, PuncToken, RelopToken, AddopToken, MulopToken, IdToken, NumToken, and EofToken.

2. Complete the Buffer class so that it has a public method to return the next character from the buffer (char next char()), and a public method to push a character back into the buffer (void unread char(char c)). Your buffer should not contain anymore than MAX BUFFER SIZE characters at any time. When designing your buffer, you may assume that the unread char() method is never called more than once without an intervening call to next char().

3. Complete the class called Scanner that implements the lexical analyzer. It should have a public method Token *next token(). Each time the function is called, it should return a pointer to a correctly initialized object representing the next lexeme in the input stream.

You should do your development on the departmental server \ice.truman.edu". I will use this platform to grade your projects.

How your project will be evaluated:
I will grade your project usingthe following criteria:
_ Quality, organization and readability of code, including comments.
_ Reasonableness of algorithms employed.
_ Adherence to input and output specifications.

What to turn in: When you are ready to turn in your project, copy all of the source files required to build your lexical analyzer (including headers, implementation files, the make file and driver program, but not the executables) into a new directory. Then, upload these files through the Blackboard tool for the assignment.