NJIT New Jersey's Science & Technology University	**CS 280: Program #2** Fall 2015 Due October 19

It's sometimes useful to make a little language for a simple problem. We are making a language to let us play with strings, and for this assignment we are building a lexical analyzer for this simple language.

Here are the lexical rules for the language:

1. The language has identifiers. An identifier starts with a letter and is followed by zero or more letters.
2. The language has string constants. A string constant is a sequence of characters, all on one line, enclosed in double quotes.
3. The language has 2 operators: + for set union and ^ for set intersection
4. The language has 4 keywords: "set", "print", "search" and "for"
5. Statements in the language end in a semicolon
6. The language supports parentheses

White space is used to separate tokens and lines for readability.

A comment begins with two slashes (//) and ends at a newline.

The lexical analyzer is to be implemented in a C++ function. The function will be passed a pointer to an input stream to read from. It will need to return the token that has been recognized and the lexeme for that token.

The definitions for the unique values for each of the tokens that you must recognize is provided in the header file p2lex.h, which is on the course website. The lexical analyzer will ignore white space and comments.

Your program will ignore whitespace and use it to note separation between tokens. The program should maintain an external integer named "linenum", which should be incremented whenever a newline is seen by the lexer.

A lexical error should cause the token ERR to be returned. An end of file should cause DONE to be returned.

You **MUST** use the p2lex.h header file for your assignment. **You may not change it.** You do not need to hand in p2lex.h.

For this assignment you should produce and hand in these two files:

- a source file that #includes p2lex.h and implements the getToken() function
- a source file that #includes p2lex.h and implements a main program to test the lexical analyzer.

Your main program should meet the following specifications:

1. The program should take an optional command line argument containing the name of an input file.
2. The program should read from the standard input if no command line argument is passed.
3. The program should read input by repeatedly calling getToken() until it returns ERR or DONE.
4. The program should keep a counter of the number of times each token was seen. At the end of the input, print out the three most common tokens and the number of times that those three tokens were seen.
5. The program should print out each unique ID that it finds after printing out the counters.