COSC 4316 Programming Assignment #2

Due Date: March 20, 2017

Problem Description:

Write a recursive descent compiler for the following grammar:

```
<program> →     BEGIN <stmt_list> END
<stmt> →         <id> := <expr> | ε
<stmt_list> →   <stmt_list> ; <stmt> | <stmt>
<expr> →        <expr> + <term> | <expr> - <term> |  <term>
<term> →        <term> * <factor> | <term> DIV <factor> | <term> MOD <factor> |  <factor>
<factor> →      <id> | <num> | ( <expr> )
```

Here, <id> represents any valid sequence of characters and digits starting with a character; and <num> represents any valid integer literal. “^” is the exponentiation operator.

Since <program> is the start symbol, you may assume that a valid program consists of zero or more statements enclosed by “BEGIN” and “END”. You will, of course, find it necessary to remove left recursion in certain productions.

The output of your compiler will be assembly code for the abstract stack machine you used in programming assignment #1. Obviously, you will only be using only a small subset of the opcodes. You’ll have to make two passes through the code; one to generate the symbol table in order build the data section of your output .asm file and the second to generate the code section.

You are to write a lexical analyzer function yylex() using lex which you can then call from your C program.

On errors, a “meaningful” error message should be printed; i.e., “invalid token in line xx”, “expression expected”, *etc*. Management would be impressed if you attempted to recover from an error rather than simply halting compilation.

Submit your source files for your compiler and lexical analyzer, along with documentation on how to build and execute your compiler. You should, of course test your solution by sending the output to your assembler and validating that the code produced is correct.