ASSIGNMENT 4

This programming assignment is about designing a scanner and a top down parser for a small scale language called XMicro. This language is an extension of the language discussed in the class. Your assignment is to extend and modify the scanner and the parser discussed in the class, so that the new scanner and the parser will scan and parse the XMicro language programs.

The XMicro language has the following lexical and syntax rules.

A program begins with the word main followed by a sequence of statements within { } block. The { } block has at least one statement. There are five types of statements. They are assignment statement, read statement, write statement, if-else statement, and while statement.

An identifier consists of lower or upper case letters and digits. An identifier begins with a letter. The identifiers are case sensitive. The following identifiers are considered reserved words with predefined meaning: main, read, write, if, else, while.

An arithmetic expression consists of variables, integer literals, plus operator +, minus operator -, multiplication operator *, division operator /, left parenthesis (, and right parenthesis). The usual arithmetic rules hold. The multiplication and division have precedence over addition and subtraction. The parentheses have higher precedence. The integer literals are positive integers or zero.

An assignment statement has a variable on the left side and an arithmetic expression on the right side of the assignment operator. The assignment operator is := The assignment statement ends with a semicolon.

A read statement has the read word followed by a list of variables within parentheses ( ) separated by commas. The read statement ends with a semicolon.

A write statement has the write word followed by a list of variables within parentheses ( ) separated by commas. The write statement ends with a semicolon.

A boolean expression consists of a left operand, a relational operator, and a right operand in that order. An operand is either a variable or an integer literal. The relational operators are <, >, <=, >=, ==, and != with their usual meanings.

An if-else statement has the if word followed by a boolean expression within parenthesis ( ) followed by a sequence of statements within { } block followed by the else word followed by a sequence of statements within { } block. The { } block has at least one statement. The statements inside the { } block can be any of the five statement types.

An if-else statement may have no else part. In that case, the if statement has the if word followed by a boolean expression within parenthesis ( ) followed by a sequence of statements within { } block.

A while statement has the while word followed by a boolean expression within parenthesis ( ) followed by a sequence of statements within { } block. The { } block has at least one statement. The statements inside the { } block can be any of the five statement types.

A comment begins with // and ends at the end of the line. The comments are ignored by the scanner and the parser.

Determine all possible token types in the XMicro language and decide how to scan and extract each token type. Your scanner will be based on these token types. Submit the complete list of token types and indicate the total number of token types.

Write the complete set of EBNF grammar rules that describe the syntax of the XMicro language. Your parser will be based on these grammar rules. Submit the complete set of EBNF grammar rules.

Your program has three components namely scanner, parser, and main. The whole program must be in one source file. The main program gives the user two options.

The first option is to scan the source code and write the sequence of tokens to another output file. For this option, the program asks the user for the source code file and the output file. If there are lexical errors then errors are printed to the screen.

The second option is to parse the source code. For this option, the program asks the user for the source code file. If there are no lexical errors and no syntax errors then a message indicating that parsing is successful is printed to the screen. Otherwise error messages are printed to the screen.

Using option one, test your program with various XMicro programs to make sure that the scanner produces correct token sequence for valid programs and produces errors for invalid programs. Using option two, test your program with various XMicro programs to make sure that the parser recognizes valid programs and produces errors for invalid programs.

The program must be written in C language. The source file must be named compiler.c. The program must compile under gcc compiler at the command line. The program must run at the command line. The program must be well documented with comments explaining all steps.