Project 2: Arithmetic Expressions and Precedence
Due: 11:59PM, Friday, October 29, 2004.

Project Objective:
1. Know the order of operations (precedence) with arithmetic operators
2. Populate and manipulate data in arrays
3. Practice use of loops (while, do..while, for)
4. Practice use of if..else statements

Project Description:
You will implement a program which takes an arithmetic expression and inserts parentheses to explicitly delineate the order of operations, as the result of the expression is dependent on it. For example, an input of 3+5*2 will return (3+(5*2)). The input will consist of single-digit integers (i.e. 0,1,2,...,9) and arithmetic operators (+, -, *, /, %). The maximum length of an inputted expression is 20; however, you must write the program using the preprocessor defined value MAX_LENGTH_OF_EXPRESSION. This will make your program more scalable. The project has two parts:

1. 1. Print out welcome message and gather user input. The welcome message must read exactly as follows:

   -------------------------------------------------------------
   Welcome to the Arithmetic Expression Evaluator!
   -------------------------------------------------------------
   Give me an arithmetic expression and I will insert the appropriate parentheses per the C language's arithmetic operator precedence rules.

Expression:

The user's input should be stored into a character array. Make sure your array is large enough to store the expression with parentheses inserted.

2. Output the final expression and intermediate steps. For example, the expression 5+3/-2 will output the following:

Expression: 5+3/-2

5+3/(-2)
5+(3/(-2))
(5+(3/(-2)))

Note: While the resulting final expression will always be the same, the intermediate steps may be different, depending on your algorithm. In any case, the number of steps will be the same. Only print one per line. We have supplied a sample algorithm; however, we encourage you to try and devise your own.
Sample Algorithm:

1. Insert user input into an array.

\[ 5 + 3 / - 2 \]

2. Determine the first two operators in the statement.

\[ 5 + 3 / - 2 \]

3. Determine the precedence of each operator.

4. If the second operator has greater precedence than the first, compare the second operator with the next operator until you find an operator with lesser precedence or discover the end of the array.

\[ 5 + 3 / - 2 \]

\[ 5 + 3 / - 2 \]

5. Insert parentheses around the sub-expression, in this instance "-2".

\[ 5 + 3 / (-2) \]

6. Repeat 2-5. Realize that you may now ignore any operators within parentheses, as a parenthesized sub-expression should now be considered as a single operand.

Project Requirements:

1. You must program using C under the Glue system and name your program `p2.c`.
2. The expression must be stored in a character array, which is scalable.
3. Your program must be properly documented.
4. Adhere to the output guidelines set in the Project Description section.
5. Submit your source `p2.c` electronically before the due date. Everyone is responsible for submitting successfully using the `sub114` command. Inability to do so is not a valid excuse for late submissions.

Grading Criteria:

Correctness: 80%
Good coding style: 10%
Proper documentation: 10%