COSC 4340 - COSC 5340
Pascal Programming Assignment
DUE DATE: Thursday, 28 September 2023

This programming assignment emphasizes Pascal language provisions for nested procedures and block structure (in the imperative paradigm context), parameter transmission by reference, alternation and iteration structures, file input/output, set operations, and recursion.

Write a complete Pascal program which will input from a file a series of infix expression character strings involving addition and multiplication on the set of single-digit integer operands and then evaluate and output result of each expression to a file. An *example* of each output line appears below:

```
THE VALUE OF 6+9*(5*(3+4)) IS 321
```

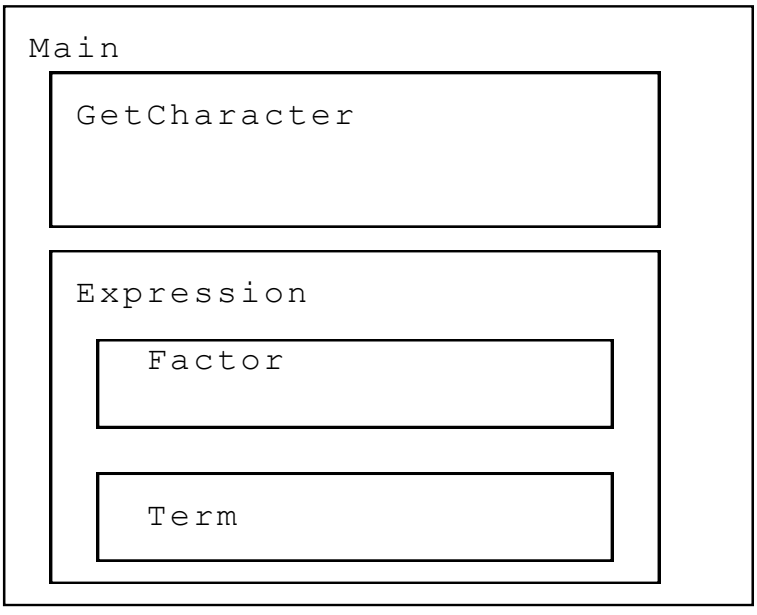
Assume that each input expression is a valid expression. The expression strings below should constitute the test data for your program (graduate students must create at least five additional expression strings to further exercise the algorithm). The input data should be organized one expression per line ... see/use the provided input file.

```
5               5*(3+4)             7*3+5*6
(5)             3+4*5+6             1*2*3*4*5*6*7
3+4             2*(((4+2)))         (((((5)))))
3*5             6+9*(5*(3+4))       1+2+3+4+5
```

The following BNF grammar describes the recursive structure of expressions involving addition and multiplication on the set of single-digit integer operands:

```
FILE ::= { LINE } <eof>
LINE ::= EXPRESSION <eoln>
EXPRESSION ::= TERM { '+' TERM }
TERM ::= FACTOR { '*' FACTOR }
FACTOR ::= digit | ( '(' EXPRESSION ')' )
```

Your program will require several *nested* **procedure**s, as diagrammed below. The types of parameters, basic logical design of each procedure, and program organization are given on the following page. Procedure **forward** declarations are not permitted. Be sure to follow the techniques of good programming style and use extensive comments to provide for internal documentation of your source program. For evaluation of this programming assignment, you will be required to separately provide your source program file, input file, and resulting output file via Canvas submission. Please submit these deliverables on or before the assignment due date.



```
program Main
{ global variables: CurrentCharacter, a character; ExprValue, an integer }

procedure GetCharacter( Token { a character passed by reference } )
      if (end-of-line)
            Read Token (i.e., eoln character) from input file
            Assign '@' to Token
      else
            repeat ( until Token is an allowable character or end-of-line )
               Read Token from input file
               Write Token to output file
            end repeat
            if ( Token is not an allowable character )
                  Assign '@' to Token
            end if
      end if
end procedure GetCharacter

procedure Expression( CurrentCharacter {a character passed by reference},
                      ExprValue { an integer passed by reference } )
{ local variable:  TermValue, an integer }

procedure Factor( CurrentCharacter { a character passed by reference },
                  FactorValue { an integer passed by reference } )
{ local variable:  Value, an integer }
      if ( CurrentCharacter is a digit )
            Assign the numerical value of the character digit to FactorValue
            GetCharacter( CurrentCharacter )
      else if ( CurrentCharacter IS EQUAL TO '(' )
                  GetCharacter( CurrentCharacter )
                  Expression( CurrentCharacter, Value )
                  Assign Value to FactorValue
                  if ( CurrentCharacter IS EQUAL TO ')' )
                        GetCharacter( CurrentCharacter )
                  end if
            end if
      end if
end procedure Factor

procedure Term( CurrentCharacter { a character passed by reference },
                TermValue { an integer passed by reference } )
{ local variable:  FactorValue, an integer }
      Factor( CurrentCharacter, FactorValue )
      Assign FactorValue to TermValue
      while ( CurrentCharacter IS EQUAL TO '*' )
         GetCharacter( CurrentCharacter )
         Factor( CurrentCharacter, FactorValue )
         Assign the product of TermValue and FactorValue to TermValue
      end while
end procedure Term

      Term( CurrentCharacter, TermValue )
      Assign TermValue to ExprValue
      while ( CurrentCharacter IS EQUAL TO '+' )
         GetCharacter( CurrentCharacter )
         Term( CurrentCharacter, TermValue )
         Assign the sum of ExprValue and TermValue to ExprValue
      end while
end procedure Expression

      while (not end-of-input-file)
         Output "THE VALUE OF "
         GetCharacter( CurrentCharacter )
         Expression( CurrentCharacter, ExprValue )
         Output " IS ", ExprValue
      end while
end program Main
```