CS 280
Fall 2025
Recitation Assignment 7

Handling Mixed Mode Expressions in BPL Language Interpreter

November 25, 2025
Due Date: December 3rd, 2025
Total Points: 6

In this assignment, you are given the definition of a class, called *Value*, which represents values of operands in the Basic Perl-Like (BPL) language interpreter (PA 3). The class represents the types of operands in BPL language (i.e., numeric, string and Boolean) using data members of C++ types as double, string, and Boolean, respectively. The objective of defining the *Value* class is to facilitate constructing an interpreter for the language which evaluates expressions and executes statements using C++.

The *Value* class includes as member functions overloaded operator functions for the evaluation of BPL arithmetic operators (+, -, *, /, and %), equality and relational operators (==, <, and >=), and logical operators (&&, ||, and !). It includes also member functions for the BPL operators: Exponentiation (Exponent), Concatenation (Catenate), string repetition (Repeat), and String relational operators (@eq, @le, @gt).

In RA 7, you are required to implement some of the member functions of the *Value* class that are used to represent the evaluation of operations by those operators. The objective of the assignment is to enable testing the class separately as a unit before using it in the construction of the interpreter in PA 3. You are required to implement the following member functions:

`Operator*(), operator<(), Catenate(), Repeat(), and SEQ()`

The semantic rules governing the evaluation of mixed-mode expressions in the BPL language are summarized below:

1. The binary operations for numeric operators are the addition, subtraction, multiplication, division, remainder (modulus), and exponentiation. These are performed upon two numeric operands. Except of the exponentiation operator, if any of the operands is a string, it will be automatically converted to a numeric value. For the remainder operator, a numeric operand is converted to an integer value in order to perform the operation.
2. The string concatenation operator is performed upon two string operands. If one of the operands is not a string, that operand is automatically converted to a string.
3. The string repetition operator must have the first operand as a string, while the second operand must be of a numeric type of an integer value.
4. Similarly, numeric relational and equality operators (==, <, and >=) operate upon two numeric type operands. While, string relational and equality operators (@eq, @le, @gt) operate upon two string type operands. The evaluation of a relational or an equality expression, produces either a true or false value. If one of the operands does not match the type of the operator, that operand is automatically converted to the type of the operator. For all relational and equality operators, no cascading is allowed.
5. Logic binary operators (&&, ||) are applied on two operands, which their Boolean values are determined based on the following rules:

- If the value is a number, 0 means false; all other numbers mean true.
- Otherwise, if the value is a string, the empty string (") and the string '0' mean false; all other strings mean true.
- If the variable doesn't have a value yet, it's an undefined variable error.

6. The unary sign operators (+ or -) are applied upon unary numeric operand only. While the unary *not* operator (!) is applied upon a *Boolean* operand, according to the rules given previously.

Note: It is recommended to implement the other overloaded operators in the *Value* class and testing them before using the class implementation in the PA 3 interpreter project.

Vocareum Automatic Grading

- A driver program, called "RA7prog.cpp", is provided for testing the implementation on Vocareum. The "RA7prog.cpp" will be propagated to your Work directory, along with the class interface of the *Value* class in the "val.h" file.
- You are graded based on 5 test cases that are generated by the driver program. The expected output results of those test cases are provided in the "RA 7 Test Cases.zip" archive and associated with the Recitation Assignment 7 on Canvas. Each test case checks the implementation of one of the required member functions to be implemented. Vocareum automatic grading will be based on the produced outputs of your implementations for the required member functions compared with the test cases output file. You may use them to check and test your implementation.
- "RA7prog.cpp" is available with the other assignment material on Canvas. Review the driver program for implementation details.

Submission Guidelines

- Please upload your implementation to Vocareum as a "val.cpp" file. The file should include the implementations of the *Value* member functions: **`Operator*(), operator<(), Catenate(), Repeat(), and SEQ().`**
- **An extended submission period of two days after the due date of your section are accepted with a fixed penalty of 25% from your score. No submission is accepted after Thursday 11:59 pm, December 5, 2025.**

Grading Table

Item	Points
Compiles Successfully	1
Test case 1: Multiplication operation	1
Test case 2: Less Than operation	1
Test case 3: String Concatenation operation	1
Test case 4: String Equality operation	1
Test case 5: String Repetition operation	1
Total	6