CMSC 430 Project 3

The third project involves modifying the attached interpreter so that it interprets programs for the complete language.

You may convert all values to double values, although you can maintain their individual types if you wish.

When the program is run on the command line, the parameters to the function should be supplied as command line arguments. For example, for the following function header of a program in the file `text.txt`:

```
function main a: integer, b: integer returns integer;
```

One would execute the program as follows:

```
$ ./compile < test.txt 2 4
```

In this case, the parameter `a` would be initialized to `2` and the parameter `b` to `4`.

An example of a program execution is shown below:

```
$ ./compile < test.txt 2 4

   1  function main a: integer, b: integer returns integer;
   2     c: integer is
   3          if a > b then
   4              a rem b;
   5          else
   6              a ** 2;
   7          endif;
   8  begin
   9      case a is
  10          when 1 => c;
  11          when 2 => (a + b / 2 - 4) * 3;
  12          others => 4;
  13      endcase;
  14  end;

Compiled Successfully

Result = 0
```

After the compilation listing is output, the value of the expression which comprises the body of the function should be displayed as shown above.

The existing code evaluates some of the arithmetic, relational and logical operators together with the reduction statement. You are to add the necessary code to include all of the following:

- All additional arithmetic operators

- All additional relational and logical operators
- Both `if` and `case` statements
- Functions with multiple variables
- Functions with parameters

This project requires modification to the bison input file, so that it defines the additional the necessary computations for the above added features. You will need to add functions to the library of evaluation functions already provided in `values.cc`. You must also make some modifications to the functions already provided.

You are to submit two files.

1. The first is a `.zip` file that contains all the source code for the project. The `.zip` file should contain the flex input file, which should be a `.l` file, the bison file, which should be a `.y` file, all `.cc` and `.h` files and a `makefile` that builds the project.
2. The second is a Word document (PDF or RTF is also acceptable) that contains the documentation for the project, which should include the following:
    a. A discussion of how you approached the project
    b. A test plan that includes test cases that you have created indicating what aspects of the program each one is testing and a screen shot of your compiler run on that test case
    c. A discussion of lessons learned from the project and any improvements that could be made

Grading Rubric

Criteria	Meets	Does Not Meet
Functionality	**70 points**	**0 points**
	Subtraction operator evaluated correctly (5)	Subtraction operator not evaluated correctly (0)
	Division operator evaluated correctly (5)	Division operator not evaluated correctly (0)
	Remainder operator evaluated correctly (5)	Remainder operator not evaluated correctly (0)
	Exponentiation operator evaluated correctly (5)	Exponentiation operator not evaluated correctly (0)
	Additional relational operators evaluated correctly (5)	Additional relational operators not evaluated correctly (0)
	Additional logical operators evaluated correctly (5)	Additional logical operators not evaluated correctly (0)
	if conditional expressions evaluated correctly (10)	if conditional expressions not evaluated correctly (0)
	case conditional expressions evaluated correctly (10)	case conditional expressions not evaluated correctly (0)
	Functions with multiple variables evaluated correctly (10)	Functions with multiple variables not evaluated correctly (0)
	Functions with parameters evaluated correctly (10)	Functions with parameters not evaluated correctly (0)
Test Cases	**15 points**	**0 points**
	Includes test cases that test all arithmetic operators (3)	Does not include test cases that test all arithmetic operators (0)
	Includes test cases that test all relational operators (3)	Does not include test cases that test all relational operators (0)
	Includes test cases that test all logical operators (3)	Does not include test cases that test all logical operators (0)
	Includes test cases that test both conditional expressions (3)	Does not include test cases that test both conditional expressions (0)
	Includes test cases with variables and parameters (3)	Does not include test cases with variables and parameters (0)
Documentation	**15 points**	**0 points**
	Discussion of approach included (5)	Discussion of approach not included (0)
	Lessons learned included (5)	Lessons learned not included (0)

	Comment blocks with student name, project, date and code description included in each file (5)	Comment blocks with student name, project, date and code description not included in each file (0)