CSCE 240 – Programming Assignment Two

Due: 11:59pm on Tuesday, September 13

Purpose

Write, test, and use the four functions described below to read a text file containing arithmetic expressions with a single operator and two operands, and output the arithmetic expression evaluated and written in words.

Functions

Function 1 – IsInteger

Write an *IsInteger* function that takes a double argument and returns true if the value truncated as an integer is equal to the original value, and false otherwise.

Function 2 – IsArithmeticOperator

Write an *IsArithmeticOperator* function that takes a character argument and returns true if the argument is one of the valid C++ arithmetic operators (+, -, *, /, %) and false otherwise.

Function 3 – OperatorToWords

Write an *OperatorToWords* function that takes a character argument and returns the following corresponding string if the character is one of the five arithmetic operators: plus, minus, times, divided by, modulo. If the character argument is not an arithmetic operator, the function will return the string “??”.

Function 4 – Compute

Write a *Compute* function that takes the following arguments: a double value for the left operand, a character value for the arithmetic operator, a double value for the right operand, and a double variable that will get the result of the operation. The function should compute the value of the operation left operand operator right operand and place the result in the fourth parameter. If the character argument is not a valid C++ arithmetic operator, the result should be set to zero and the function should return false. If the arithmetic operation is invalid (dividing by zero or modular division with non-integers), the result should be set to zero and the function should return false.

For example

Compute(3.1, ‘*’, 4, answer)	should return true, and answer should be set to 12.4
Compute(5, ‘%’, 3, answer)	should return true, and answer should be set to 2
Compute(5, ‘%’, 3.2, answer)	should return false, and answer should be set to 0
Compute(2.5, ‘^’, 3, answer)	should return false, and answer should be set to 0

Program 2

Your *main* function should read the file *arithmetic_expressions.txt*, which holds arithmetic expressions, and output the results of the arithmetic expressions in words to the standard output device (using cout) as shown below. The main function must call the *Compute* function you've implemented in *operator_functions.cc* to assist with this task.

The output for any expression including an invalid arithmetic operator should be "Unrecognized operation *symbol*". The output for an invalid operation should be "Could not compute *expression*". See the sample input/output pairs given below.

Example Input/Output Pair

Input: 5.25 / 2.1

Output: 5.25 divided by 2.1 is 2.5

Example Input/Output Pair

Input: 3.7 / 0

Output: Could not compute 3.7 / 0

Example Input/Output Pair

Input: 7.2 ^ 4.1

Output: Unrecognized operation ^

Specifications

- All output should be directed to the standard output device using cout.
- The prototypes for the *IsInteger*, *IsArithmeticOperator*, *OperatorToWords*, and *Compute* functions must be included in *operator_functions.h*
- The *IsInteger*, *IsArithmeticOperator*, *OperatorToWords*, and *Compute* functions must be implemented in *operator_functions.cc*
- The *main* function must be implemented in *program2.cc*
- You will submit a zip file (only a zip file will be accepted) containing *operator_functions.h, operator_functions.cc,* and *program2.cc* to the assignment in Blackboard.
- Source files must compile and run on a computer of the instructor's choosing in the Linux lab (see your course syllabus for additional details).

Testing

Initial unit tests have been included for the functions in the attached files *test_isinteger.cc*, *test_isarithmeticoperator.cc*, *test_operatortowords.cc*, and *test_compute.cc*. You should ensure that your functions pass the included tests, and you are encouraged to create more rigorous tests. Your functions will be graded using unit tests similar to the included tests, with different values.

A python script, *test_program2.py*, has been provided for you to test your program2 source file with the included *arithmetic_expressions.txt* file. In order to use the tester, you'll need access to a python3 interpreter. Ensure that the tester, *arithmetic_expressions.txt*, and your *program2* executable are in the same directory, and python3 is in your path. The command to run the tester is given below:

```
python3 test_program2.py
```

program2 will be graded using a similar python script with a different *arithmetic_expressions.txt* input file and the corresponding expected output.

Grade Breakdown

Style: 1 point

Documentation: 1 point

Clean compilation: 1 point

IsInteger passes instructor's unit tests: 1 point

IsArithmeticOperator passes instructor's unit tests: 1 point

OperatorToWords passes instructor's unit tests: 1 point

Compute passes instructor's unit tests: 2 points

program2 runs correctly with instructor's test data set 1: 2 points

The penalty for late program submissions is 10% per day, with no program being accepted after 3 days.