Calculator List

COP 4530 Programming Project 1

Instructions

For Programming Project 1, you will implement a linked list based arithmetic calculator. The calculator will be able to perform addition, subtraction, multiplication, and division. The calculator will keep a running total of the operations completed, the number of operations completed, and what those operations were. The calculator will also have an "undo" function for removing the last operation. The calculator will also be able to output a string of the operations completed so far with fixed precision.

The calculator (which must be called "CalcList") has to be implemented using a singly, doubly, or circularly linked list. Any projects that use the C++ Standard Library Lists or other sources to implement the linked list will receive a zero. The calculator has to implement at least four methods:

Abstract Class and Files

double total() const

This method returns the current total of the CalcList. Total should run as a constant time operation. The program should not have to iterate through the entire list each time the total is needed.

void newOperation(const FUNCTIONS func, const double operand)

Adds an operation to the CalcList and creates a new total. The operation alters total by using the function with the operand. Example: newOperation(ADDITION, 10) => adds 10 to the total.

void removeLastOperation()

Removes the last operation from the calc list and restores the previous total.

std::string toString(unsigned short precision) const

Returns a string of the list of operations completed so far formatted with a fixed point precision. The form of the string should strictly be: "(step): (totalAtStep)(Function)(operand) = (newTotal)\n".
Example: toString(2) => "3: 30.00*1.00=30.00\n2: 10.00+20.00=30.00\n1: 0.00+10.00=10.00\n"

This project includes an abstract class for the CalcList from which to inherit. This abstract class (CalcListInterface) contains the pure virtual version of all the required methods. This file also includes a typedef of an enum used for the four arithmetic functions called FUNCTIONS.

This project will be tested using the Catch2 (https://github.com/catchorg/Catch2) test framework. This framework only requires that a program include the header to run the test file. The test file that will be used to grade the project is included and can be used to test your code before submitting. For your convenience, this project contains the Catch2 header. The course documents include a quick tutorial on how to use Catch2 and test files.

Please submit complete projects as zipped folders. The zipped folder should contain: PP1Test.cpp, catch.hpp, CalcListInterface.hpp, CalcList.hpp (Your Code), CalcList.cpp (Your Code)

Examples

Below are some examples of how your code should run. The test file can also be used to get an idea of how the code should run.

```
CalcList calc;                    // Total == 0
calc.newOperation(ADDITION, 10);      // Total == 10
calc.newOperation(MULTIPLICATION, 5); // Total == 50
calc.newOperation(SUBTRACTION, 15);   // Total == 35
calc.newOperation(DIVISION, 7);       // Total == 5
calc.removeLastOperation();           // Total == 35
calc.newOperation(SUBTRACTION, 30);   // Total == 5
calc.newOperation(ADDITION, 5);       // Total == 10
calc.removeLastOperation();           // Total == 5

// Should Return:
// 4: 35.00-30.00=5.00
// 3: 50.00-15.00=35.00
// 2: 10.00*5.00=50.00
// 1: 0.00+10.00=10.00
std::cout << calc.toString(2);

calc.removeLastOperation();           // Total == 35

// Should Return:
// 3: 50-15=35
// 2: 10*5=50
// 1: 0+10=10
```

```
std::cout << calc.toString(0);
```

Hints

When implementing the toString method, the headers sstream and iomanip will have functions that make controlling precision and creating returnable string easier. Also remember, zero does not have a multiplicative inverse.

Rubric

Any code that does not compile will receive a zero for this project.

Criteria	Points
Total should be initially zero	1
Operations should be removable	5
Zero multiplication operation should be removable	2
Operations cannot divide by zero and should throw	2
Removal of operations from an empty CalcList should throw	2
toString functions should return string list of operations at precision	5
Operations should change the total	10
Calculator uses a student implemented Linked List	7
Code uses object oriented design principles (Separate headers and sources, where applicable)	3
Code is well documented	3
Total Points	**40**