Programming Assignment One

A hallmark of any devoted computer science geek is that he or she will spend an inordinate amount of time creating complex computing solutions to problems that could be solved quickly and trivially using preexisting tools. With that in mind, this first programming assignment will require you to spend hours and hours writing a program that will… add two numbers… among other things, of course. Who needs an $8 calculator from Wal-Mart® when a true geek can write one in C++?

Your task is to write a calculator class in C++ that could be used by fellow coders to perform certain mathematical operations. You will not be turning in a calculator program, but rather a calculator class - you will make calculator operations available for use in programs, just as the various C++ libraries make operations available to coders. In this manner you will imitate a software vendor providing features to be included in other coders’ programs.

Requirements:
Your class definition must be written in a file called calculator.h, and your member functions must be written in a file called calculator.C. These files will be turned in; however, in order to test your class before you turn it in, you must write your own program that uses your class - this program should make a calculator object and call the various public interface functions to verify that your class gives accurate results to calculations. After you turn in your two class files, I will test them with my own program that will likewise test the interface functions.

Your calculator class must make six functions publicly available to the class user. These functions will represent mathematical calculations and will be assigned to you randomly from the difficulty list below (one from the high list, two from the mid list, and three from the low list.) Each of the six functions must meet the exact specifications below so that they will work with my test program.

Your functions only need to work with integers… specifically, only the int C++ type need be used. Your functions will not have to work with floating point numbers, nor will they have to work beyond the bounds of the int type (keep in mind the int type includes positive and negative integers.) I will not test your program with any integer greater than 1000 nor any integer less than -1000

You may not use any math library or any math functions from other libraries. The only library you may include in your class is iostream.h.

Your class must have a default constructor. Additional constructors are optional, as they may or may not make sense for your implementation. Destructors are unnecessary.

All functions should be written by you; they should not be copied from external sources. You have full leeway in how to calculate results - as long as the specifications on the functions are honored and the answers are correct, I don’t care if you use traditional mathematical algorithms or you use “shortcuts”.

You must also write a short (3-4 paragraphs) report on the issues you faced in creating the solution: problems that came, insights that came to you, and evaluations of how easy or how difficult the major parts of the project were. Also in this report you should declare any and all resources you used for the project:

Conversations: with whom and about what, including instructors and classmates
Websites: URL(s) for page(s) you researched
Textbooks: names and sections

This should be in a pure-text file called report.txt.

**High**

*Slope between two points on a Cartesian coordinate plane*

Function Name: slope
Function Parameters: integer1x, integer1y, integer2x, integer2y
Return: nothing
Screen output: The slope between \((\text{integer1x}, \text{integer1y})\) and \((\text{integer2x}, \text{integer2y})\) is answer.

Note - The answer should be written in the form of a fraction such as “2/3”. If the denominator is zero, the answer should be “undefined.” If the denominator is non-zero but the numerator is zero, the answer should be “zero.” If the numerator and denominator are both negative, the negative signs should not be written in the answer. The fraction does not have to be written in simplified form (“4/8” is acceptable.)

*Determining if an integer is prime*

Function Name: prime
Function Parameters: integer1
Return: nothing
Screen output: integer1 is/isn’t a prime number.

*Permutations of m objects taken n at a time*

Function Name: permutation
Function Parameters: integer1, integer2
Return: nothing
Screen output: There are answer ways to arrange integer1 objects in ordered groups of integer2.

**Mid**

*Factorial of an integer*

Function Name: factorial
Function Parameters: integer1
Return: nothing
Screen output: The factorial of integer1 is answer.

*Least common multiple of two integers*

Function Name: lcm
Function Parameters: integer1, integer2
Return: nothing
Screen output: The least common multiple of integer1 and integer2 is answer.

*Greatest common factor of two integers*

Function Name: gcf
Function Parameters: integer1, integer2
Return: nothing
Screen output: The greatest common factor of integer1 and integer2 is answer.

*One integer raised to the power of another*

Function Name: power
Function Parameters: integer1, integer2
Return: nothing
Screen output: \textit{integer1} raised to the power of \textit{integer2} is \textit{answer}.

\textbf{Low}

\textit{Sum of two integers}
Function Name: \texttt{sum}
Function Parameters: \texttt{integer1, integer2}
Return: nothing
Screen output: The sum of \textit{integer1} and \textit{integer2} is \textit{answer}.

\textit{Difference of two integers}
Function Name: \texttt{difference}
Function Parameters: \texttt{integer1, integer2}
Return: nothing
Screen output: The difference of \textit{integer1} and \textit{integer2} is \textit{answer}.

\textit{Product of two integers}
Function Name: \texttt{product}
Function Parameters: \texttt{integer1, integer2}
Return: nothing
Screen output: The product of \textit{integer1} and \textit{integer2} is \textit{answer}.

\textit{Opposite of an integer}
Function Name: \texttt{opposite}
Function Parameters: \texttt{integer1}
Return: nothing
Screen output: The opposite of \textit{integer1} is \textit{answer}.

\textit{Absolute value of an integer}
Function Name: \texttt{absvalue}
Function Parameters: \texttt{integer1}
Return: nothing
Screen output: The absolute value of \textit{integer1} is \textit{answer}.

\textbf{Example of the Class in Use}
Suppose I have used your calculator class to make a calculator object called \texttt{myCalculator} in my \texttt{main()}. I would call the sum function as follows:

\begin{verbatim}
    myCalculator.sum(5, 19);
\end{verbatim}

The output on the screen should be:

The sum of 5 and 19 is 24.

\textbf{Submission}
You will submit your three files - \texttt{calculator.h}, \texttt{calculator.C}, and \texttt{report.txt} - electronically using a special submission program. Details on using this program will be provided later.

\textbf{Deadline}
Your two class files must be submitted successfully by 11:59 pm on June 12, 2003.