In this assignment, we will create a calculator that perform simple arithmetic operations on complex numbers. Complex values are denoted by a parenthesized pair of values separated by a comma representing the real and imaginary part of the variable. For example \((1, 2)\) indicated the real part is 1 and the imaginary part is 2. A complex number can also be represented by the magnitude and angle format like this \((1 > 45)\) indicating a complex value with a magnitude of 1 and an angle of 45 degrees. Finally, a single numeric value without parenthesis indicates a complex number with the real value equal to the specified numeric value and an imaginary part of zero.

There are three basic requirements for the calculator.

1. If the input line has two operands and one operator, the calculator should perform the specified operation and print the result. Further, the result should be saved for use in requirement 2 below. An example is:

\[(1,2) + (2,3) \quad \text{(input by user)} \]
\[= (3,5) \quad \text{(output by the program)} \]

2. If the input line starts with an operator followed by a single operand, the result from the prior operation is used as the left–side operand. An example is:

\[(1,2) + (2,3) \quad \text{(input by user)} \]
\[= (3,5) \quad \text{(output by the program)} \]
\[+ (5,5) \quad \text{(input by user)} \]
\[= (8,10) \quad \text{(output by the program)} \]

3. Finally, a single operand with no operator simply outputs the value of the single operand and stores it as the prior result.

\[(1,2) \quad \text{(input by user)} \]
\[= (1,2) \quad \text{(output by the program)} \]
\[+ (5,5) \quad \text{(input by user)} \]
\[= (6,7) \quad \text{(output by the program)} \]

In order to complete this assignment, your program must be written in C++ and implement the functions described in the skeleton code provided. You will need to implement the Complex class, and provide overloaded operators for the plus, minus, multiply, and divide operators. You will NOT need an exponentiation operator for this assignment. The Complex class will need a constructor with no arguments (default constructor), one with two arguments with initial values of both the real and imaginary part, and a third constructor that builds a complex number from a const string\& string in the format described below. You will also need a member function to calculate the magnitude of the complex value, the angle of the value, and the complex conjugate of the value. Finally, you will overload the << operator to provide a printing function.
To parse the input strings for your complex calculator, a parser is provided. The parser takes an input line as input, and breaks it into substrings separated by numeric operators (+, -, * and /). Documentation of how to call the parser is included in the provided source code. If the first character of a substring is a left parenthesis, then all characters up to and including the subsequent right parenthesis is parsed as part of the substring. Thus the input:

\[(1, 2) + 3\]

would be parsed as two substrings, the first is \((1, 2)\) with a “+” delimiter, and the second is the substring 3.

It is possible that a complex variable is Not a Number (NaN). This is the case when it is the result of a divide by zero, when it is constructed from an ill–formatted string, when computing the angle of a zero magnitude complex value, or when the result of any operation where either of the operands are NaN. When printing a complex value that is NaN, the string "NaN" should be output.

A sample session is shown below:

\[(102, 0) - (1, 0)\]
\[= 101\]
\[(1, 2) + (3, 6)\]
\[= (4, 8)\]
\[+ (2, 3)\]
\[= (6, 11)\]
\[+ (0, 1)\]
\[= (6, 12)\]
\[/ (2, 0)\]
\[= (3, 6)\]
\[+ 5\]
\[= (8, 6)\]
\[(1 > 45)\]
\[= (0.707107, 0.707107)\]
\[/ 0\]
\[= \text{NaN}\]
\[+ 10\]
\[= \text{NaN}\]
\[1\]
\[= 1\]
\[*(100, 23)\]
\[= (100, 23)\]

There are a few more things you need to know.

1. The value of \(\pi\) is defined in “math.h” as the symbol \(\text{M_PI}\).

2. The trig functions you need are also defined in “math.h”. In particular are \(\sin\), \(\cos\) and \(\text{atan2}\). All trig functions use radians, not degrees. The \(\text{atan2}(y, x)\) is \(\text{arc-tangent}\). Since tangents can legally be infinity, \(\text{atan2}\) uses the form \(y/x\) and passes both \(y\) and \(x\) as arguments, getting around the possible problems with infinite values.

3. The complex conjugate of a complex variable is another complex variable (r, -i). In other words, just negate the imaginary part.

4. The magnitude of a complex variable is just the square root of the real squared plus the imaginary squared.

5. To compute \(a/b\) where both \(a\) and \(b\) are complex values, do the following:

   (a) If the magnitude of \(b\) is zero, the result is \(\text{Not-a-Number}\).

   (b) Compute a temporary variable \(c\) as the \(a\) times the complex conjugate of \(b\).

   (c) Compute \(m\) as the magnitude of \(b\) squared.

   (d) The real part of the result of the division is \(c.\text{real}/m.\text{real}\).

   (e) The imaginary part of the result is \(c.\text{imag}/m.\text{real}\.

2
**Resources.** Included on the class web page are:

1. A skeleton `complex-calc.cc` program that is a starting point. It contains the subroutine to read a line from standard in and call the string parser. Your main loop should exit when an empty line is encountered.

2. Skeleton `complex.h` describes the requirements for the `Complex` class. You should implement the required functions in `complex.cc`.

3. `string-parse.h` and `string-parse.cc` are provided for string parsing.

4. A `Makefile` for building the executable binary.

Your program can be compiled and tested on any available computing platform that has a C++ compiler. The instructor will compile and test your program on the GTL linux systems. Be sure to put your name on the source code in the comments section.

**Turning in your Project.** The assignment is to be turned by via EMail to George Riley riley@ece.gatech.edu. The Subject: line must say ECE3090 Program 1 Submission. Include as attachments your `complex.cc` and your main program in `complex-calc.cc`. I will compile your program on the linux platform and test it with several legal operations.