ST. CHARLES COMMUNITY COLLEGE

CPT-182-83/R83 (23/SP) - Programming in C++

Exam 2 (40 Points)

Deadline: Thursday, April 6, 2023, by 07:59 AM (Code Submitted on Canvas)

Project Overview

In mathematics, a *complex number* (c) has the following form:

$$c = a + bi$$

Both a and b are real numbers. a is called the *real part*; b is called the *imaginary part*. i is the *imaginary unit* (a constant value), where $i = \sqrt{-1}$.

Below are a few examples of complex numbers:

$$5 + 3i,\ -6 - 11i,\ \frac{5}{3} + \frac{1}{6}i$$

Two complex numbers, $c_1 = a_1 + b_1 i$ and $c_2 = a_2 + b_2 i$, can be added together using the '**+**' operator, which generates another complex number (see below).

$$c_1 + c_2 = (a_1 + b_1 i) + (a_2 + b_2 i) = (a_1 + a_2) + (b_1 + b_2)i$$

In this project, you are going to write a C++ program that reads in complex numbers in pairs from an input file, adds each pair of complex numbers together, and writes the results to an output file.

In this project, for simplicity, the real and imaginary parts of all the complex numbers are **integers**, instead of ~~floating-point numbers~~.

The `Complex_Number` Class

1) The class has the following data fields (variables):
   - **`real`**: an integer that represents the *real part* of the complex number
   - **`imaginary`**: an integer that represents the *imaginary part* of the complex number
2) The class has the following constructor(s):
   - A **default constructor** that sets both the real part and imaginary part to **0**

- A **constructor** that takes two integers as arguments and sets the real part and imaginary part with the values passed in
- If you like (**not** required), the two constructors mentioned above can be combined into one constructor by using *default parameter values*.

3) The class should overload the following operators.
   - Arithmetic addition operator ('**+**'). This operator function should allow two **`Complex_Number`** objects be added together.
   - Stream extraction operator ("**>>**"). The operator function should allow the program to read in a complex number from an input stream directly (e.g., "**`fin >> c_num`**" where **`fin`** is an input stream and **`c_num`** is a **`Complex_Number`** object).
   - Stream insertion operator ("**<<**"). The operator function should allow the program to write a complex number to an output stream directly (e.g., "**`fout << c_num`**" where **`fout`** is an output stream and **`c_num`** is a **`Complex_Number`** object).
4) For each operator function in this class, it is your responsibility to correctly determine the number of arguments, types of arguments, whether the function should be a **`const`** function, whether the function should be a **`static`** function, and whether the function should be a **`friend`** function. **Failing** to do these will result in **losing** points.

The Input File

1) The input file is a **plain text file** (filename: **`complex_numbers.txt`**).
2) In each row of the input file (except for those empty lines), there are **2 complex numbers** stored, which equals to **4** data fields, *real part of the first complex number*, *imaginary part of the first complex number*, *real part of the second complex number*, and *imaginary part of the second complex number* (in that order). Data fields are separated by whitespaces.
3) Please see a **sample row** of the input file below (with explanations).

3	**5**	**16**	**−3**
⬆	⬆	⬆	⬆
Real part of the 1st complex number	*Imaginary part* of the 1st complex number	*Real part* of the 2nd complex number	*Imaginary part* of the 2nd complex number

4) You **cannot** assume (or guess) the number of rows of complex numbers stored in the input file. In other words, no matter how many rows of complex numbers are stored in the input file, your program should correctly process all of them.
5) There may be empty lines at the beginning, in the middle, and/or at the end of the input file. Your program should smartly skip those empty lines and process only the rows containing data.
6) Please refer to the sample input file to better understand the input file format.

The Output File

1) The output file is a **plain text file** (filename: **`results.txt`**).
2) After adding the two complex numbers in a row in the input file, your program should write the result complex number to the output file, one result per row.
3) A **`Complex_Number`** object should be outputted as its *real part*, a tab character, then its *imaginary part* (in that order).
4) Please refer to the sample output file to better understand the output file format.

The `main()` Program

1) The **`main()`** function has already been completely written for you. Please download the file **`Main.cpp`** on Canvas.
2) The **`main()`** program processes the input file line-by-line. In each row, it reads the two complex numbers stored, adds them together, and writes the result complex number to the output file.
3) You **cannot** change anything of the written **`main()`** function. In other words, you **must** use the written **`main()`** function provided. If you write the **`Complex_Number`** class correctly, then the given **`main()`** function will work perfectly to generate correct output file.
4) When you submit the project, you **only** need to submit **two files**, "**`Complex_Number.h`**" and "**`Complex_Number.cpp`**". Do **not** submit "**`Main.cpp`**".

Deadline: Thursday, April 6, 2023, by 07:59 AM (Code Submitted on Canvas)