PIC-10A: Homework 8

Due 12/05/21 11:59pm via CCLE

Problem 1 (10pts):

Given the definition of a class ComplexNumber, implement its constructors and member functions, and non-member functions

```
class ComplexNumber {
private:
      double real, imag;
public:
      // constructor
      ComplexNumber ();
      ComplexNumber (double, double);
      ComplexNumber (const ComplexNumber &);

      // accessors
      double realPart() const;
      double imagPart() const;
      double norm() const;
      void display() const;
      double angle() const;

      // mutators
      void set_real(double);
      void set_imag(double);
      void set_value(double, double);
      void add(const ComplexNumber &);
      void subtract(const ComplexNumber &);
      void multiply(const ComplexNumber &);
      void multiply(const double);
      ComplexNumber conjugate();
};

//non-member functions
ComplexNumber add(const ComplexNumber&z1, const ComplexNumber&z2);
ComplexNumber subtract(const ComplexNumber&z1,const ComplexNumber&z2);
ComplexNumber multiply(const ComplexNumber&z1,const ComplexNumber&z2);
```

The class describes complex numbers and its member functions as follows:

- `ComplexNumber()`: default constructor that set the real and imaginary parts to be 0
- `ComplexNumber(double , double )`: constructor that sets the real and imaginary parts to be the first and second parameters respectively.
- `ComplexNumber(const ComplexNumber &)`: copy constructor that sets the real and imaginary parts to be the ones of a given ComplexNumber object.
- `double realPart() const`: returns the real part of the ComplexNumber object
- `double imagPart() const`: returns the imaginary part of the ComplexNumber object
- `double norm() const`: returns the norm of this complex number,

$$norm = \sqrt{imag^2 + real^2}$$

- `double angle() const`: returns the angle created by the current ComplexNumber and the positive real-axis. **The angle has to be in the range** $(-\pi, \pi]$. You are allowed to use functions `atan(double)` or `atan2(double, double)` from `cmath` library.
- `void display() const`: displays the complex number. For example, if `real = 3.0, imag = -2.0`, display `"3-2i"`
- `void set_real(double re)`: sets the real part to be the given parameter re.
- `void set_imag(double im)`: sets the imaginary part to be the given parameter im.
- `void set_value(double re, double im)`: sets the real and imaginary parts to be the first and second parameters respectively.
- `void add(const ComplexNumber& z);` adds the parameter ComplexNumber z to the current ComplexNumber.
- `void subtract(const ComplexNumber & z)`: subtract the parameter ComplexNumber z from the current ComplexNumber.
- `void multiply(const ComplexNumber& z)`: multiply the current ComplexNumber by the parameter ComplexNumber z. The multiplication between 2 complex numbers are given by the formula

$$(x_1 + y_1 i).(x_2 + y_2 i) = x_1 x_2 - y_1 y_2 + (x_1 y_2 + x_2 y_1)i$$

- `void multiply(double scalar)`: scalar multiplication. Multiply the real and imaginary part of the current complex number by the parameter scalar.
- `ComplexNumber conjugate()`: returns a ComplexNumber object which is the complex conjugate of the current ComplexNumber. Recall the complex conjugate of `x+yi` is `x-yi`
- `ComplexNumber add(const ComplexNumber & z1, const ComplexNumber & z2)`: return a ComplexNumber object which equals to z1 + z2
- `ComplexNumber subtract(const ComplexNumber & z1, const ComplexNumber & z2)`: return a ComplexNumber object which equals to z1 - z2
- `ComplexNumber multiply(const ComplexNumber & z1, const ComplexNumber & z2)`: return a ComplexNumber object which equals to z1 * z2

Download the template *hw8_1.cpp*, implement the class, save, and submit it as *hw8_1.cpp*

Problem 2 (5pts):

Write a function

```
vector <int> merge (const vector<int> & a, const vector<int> & b)
```

That merges two vectors, alternating elements from both vectors. If one vector is shorter than the other, then alternate as long as you can and then append the remaining elements from the longer vector. For example, if `a` is:

1 4 9 16

And `b` is

8 7 5 8 11

Then `merge` returns the vector

1 8 4 7 9 5 16 8 11

Download the template *hw8_2.cpp*, implement the class, save, and submit it as *hw8_2.cpp*