Lab Manual – Complex Numbers with Dynamic Data

Important Note:

- **Define your class in "ComplexNumber.h" file, Implement all the functions of ComplexNumber class in "ComplexNumber.cpp" and Test your class in "Driver.cpp"**
- **Make sure that there is no memory leakage, dangling pointers and exceptions in your program**

1- Implement following **ComplexNumber** class

```
class ComplexNumber

{

      private:
         int* real;       //Integer will be saved on heap
         int* imaginary;  //Data will be saved on heap
      public:
         ComplexNumber(int, int); /*Constructor with Default arguments,
      allocate memory properly and display Constructor Called message with
      data.*/
         ~ComplexNumber();      /*Deallocate memory properly and Display
      Destructor Called message with data.*/
         void Input();    //Takes input from user
         void Output();   //Properly display Complex number like a+bi
         float Magnitude();     //Calculates magnitude of a complex number
};
```

2- Create a complex number c1 on stack using default constructor. Take c1's data using your input function. Display c1 using Output function.

3- Create a ComplexNumber pointer cPtr in your main and check if it implicitly calls Constructor or not.

4- Create a Complex Number 3+5i **on heap** and save its address in cPtr. Print the complex number using cPtr.

5- Create an array of five complex numbers on stack. Take complex numbers input from user in for loop. Display all these complex numbers in for loop, along with their magnitude. Do we need to delete this array?

6- Dynamically create an array of complex numbers after taking the size of array from user. Input and output these complex numbers using Input Output functions. Also display their magnitude. What will be the memory configuration in this case? Do we need to de-allocate anything?

7- Run following piece of code. Why does it crash? Identify functions it needs in order to run successfully. Do not add any functions in your class at this point.

```
ComplexNumber c1(3,4);
ComplexNumber c2(4,5);
{
ComplexNumber temp = c1;
C1 = c2;
C2 = temp;
}
C1.Print();
C2.Print();
```

8- Run the code given below with your class definition, why does it crash?

```
ComplexNumber c1(5,10);

{

ComplexNumber c2(c1);

Cout<<”C2 = ”;

C2.Print();

}

Cout<<”C1 = ”

C1.Print();
```

9- Write Copy Constructor for your class and run the above code again. It should run successfully. Print “Copy Constructor Called” with data.

10- Run following code with your class definition, why does it crash?

```
ComplexNumber c1(3,4);
{
ComplexNumer c2, c3;
C3 = c2 = c1;

C2.Print();
C3.Print();
}
C1.print();
```

11- Write Assignment Operator of your class and run above code again. It should run successfully.

12- Dry run following piece of code and write its output. Verify your output on console. This code should run successfully this time.

```
ComplexNumber c1(3,4);
ComplexNumber c2(4,5);
{
ComplexNumber temp = c1;
C1 = c2;
C2 = temp;
}
C1.Print();
C2.Print();
```