```cpp
 1  //
 2  // Shaun Chemplavil U08713628
 3  // shaun.chemplavil@gmail.com
 4  // C/C++ Programming III : Intermediate Programming with Objects
 5  // 151116  Raymond L. Mitchell III
 6  // Complex.cpp
 7  // Win10
 8  // Visual C++ 19.0
 9  //
10  // File contains the display member function for the SavingsAccount class
11  //
12
13  #include <iostream>
14  #include "Complex.h"
15
16  using std::ostream;
17  using std::istream;
18
19  namespace ShaunChemplavil
20  {
21     // Default Constructor
22     Complex::Complex()
23        : real(0.0), imaginary(0.0) {}
24
25     Complex::Complex(double real)
26        : real(real), imaginary(0.0) {}
27
28     Complex::Complex(double real, double imaginary)
29        : real(real), imaginary(imaginary) {}
30
31     ostream &operator<<(ostream &out, const Complex &value)
32     {
33        out << value.real;
34
35        //Check if imaginary part is positive, and place '+'
36        // (negative sign from imaginary will automatically be displayed)
37        if (value.imaginary >= 0.0)
38           out << "+";
39
40        out << value.imaginary << "i";
41
42        return out;
43     }
44
45     istream &operator>>(istream &in, Complex &value)
46     {
47        //a '+/-' separating the real from imaginary will put values in correct
48        // member variables
49        in >> value.real;
50        in >> value.imaginary;
51
52        // Ignore the character i
```

```cpp
53          in.ignore();
54
55          return in;
56      }
57
58      Complex operator+(const Complex &op1, const Complex &op2)
59      {
60          return(Complex(op1.real + op2.real, op1.imaginary + op2.imaginary));
61      }
62
63      Complex operator-(const Complex &op1, const Complex &op2)
64      {
65          return(Complex(op1.real - op2.real, op1.imaginary - op2.imaginary));
66      }
67
68      bool operator==(const Complex &op1, const Complex &op2)
69      {
70          return((op1.real == op2.real) && (op1.imaginary == op2.imaginary));
71      }
72
73      bool operator!=(const Complex &op1, const Complex &op2)
74      {
75          return(!(op1 == op2));
76      }
77  }
78
```