The text gives excellent coverage of structures (a data type with data members) and classes (a data type with data members and function members) in Chapter 6. You should find your assignment doable by reading these sections and studying the class definition provided below, together with the main program which uses the class

A complex number is one which takes the form: A + Bi (where i is a symbol, not an unknown). 'A' is called the 'real part' and 'B' is the 'imaginary part'. Complex number are used in higher mathematics, and can be added, output, compared, and negated, among other things. Addition isperformed by adding the real parts and the imaginary parts. Comparison of two complex numbers is done by comparing the real parts of the two, and the imaginary parts. Negation follows suit.

The following class Complex provides a way for a program to provide a Complex type, and work with complex numbers.

```
// CONTENTS of Complex.h header file

#include <iostream>
using namespace std;

class Complex{

public:
                                //constructors
        Complex();
        Complex(double re);
        Complex(double re, double im);

                                // accessor functions
        double get_real();
        double get_imag();

                                // math functions
        void add(Complex b);
        bool isequal(Complex second);
        void negate();

                                // output function
        void outComplex();

private:
        double real;    // real part of Complex number
        double imag;    // imaginary part of Complex number
                // data members of this class will only be manipulated by
                // the class methods
```

```
};   // don't forget this semicolon !!

// CONTENTS of Complex.cpp source code file

#include “Complex.h”
Complex::Complex(){        // constructor, places 0.0 in both members
   real=0.0;
   imag= 0.0;
}


Complex::Complex(double re){   // constructor inits real part to parameter
   real = re;
   imag = 0.0;
}


Complex::Complex(double re, double im){  //constructor inits real and imag parts
   real = re;
   imag = im;
}


double Complex::get_real(){             // way  to access 'real' part
          return real;
}


double Complex::get_imag(){             // way to access 'imaginary' part
          return imag;
}


void Complex::add(Complex b){        // adds argument to object making 'call'
           real = real + b.real;
           imag = imag + b.imag;
}

bool Complex::isequal(Complex second){     // is the argument object equal to 'caller'
    if ((real == second.real) && (imag== second.imag))
                    return true;
           else
                    return false;
}
```

```
void Complex::negate(){           // A + Bi  =>  -A - Bi
          real = -real;
          imag = -imag;
}


void Complex::outComplex(){       // output value
        cout << real << '+' << imag << "i\n";
}

//CONTENTS  of app.cpp application program file which uses the Complex class
#include “Complex.h”
void main (){
                        // three new objects of type class Complex
        Complex first;            // constructor w/ no args
        Complex second(4.2);      // constructor w/ one arg
        Complex third(9.5,7.4);   // constructor w/ both args

        double real_part, imag_part;

        first.outComplex();       //output new objects
        second.outComplex();
        third.outComplex();

        real_part = third.get_real();   //get individual parts and output
        imag_part = third.get_imag();
        cout << "real part" << real_part << " imag part" << imag_part << "\n";

        second.add(third);            // add 'third' parts to object 'second'
        second.outComplex();          //output new value of 'second'

        if (first.isequal(second))    // is first equal to second??
                cout << "they are equal\n";
        else
                cout << "they are not equal \n";

        third.negate();               // negate object 'third'
        third.outComplex();           // output updated 'third'

        return 0;
}
```