Using MS Visual C++ write the following programs. Please email your homework to Steve at sjbrown8@eng.usf.edu. You should have read chapters 10, 11, & 12 in your book. Our final is scheduled on May 5 from 8 to 10. It is not too late to ask for help from Steve & I.

**Exercises**

1) Remember your structure called Complex? Define a class called Complex that is similar to your previous structure. Make your class public. Show in a main that you can define variables of Complex type and that you can manipulate the data to those variables via the dot membership operator. Demonstrate some cin & cout also.

2) Define four functions to be added to problem #1. These functions should be add, sub, mult, & conjugate (no scalar multiply). Make these functions all internal class functions. Show that they all work properly with the appropriate main code.

3) Modify problem #2 so that your functions are external member functions.

4) Add to problem #3 a default constructor that insures that Complex objects are initialized to zero, and an overloaded constructor that accepts two items that permits the user to define a Complex object with a particular value. Each of your constructors should have a different output lines to facilitate quick understanding of when they were activated. For example: “A default constructor called” or “Constructor called”. Your program should also have a destructor with some type of output. Have your main code demonstrate each new feature.

5) Modify problem #4 so that there is only one constructor that supplies defaults in it’s argument list. Show that it is working for all possible input options…

6) Add two more functions that allow a user to input and output from Complex objects. Change your data to private. Change each of the original 4 functions to overloaded operators. Write a main program that clearly shows that each overloaded operator is correct. Use the in/out functions to insure your calculations are correct.

7) Learn about multiply overloaded operators by adding the scalar multiply to problem #6. Do you see that means adding TWO (2) new functions, one that is not a member of the Complex class?

8) Change problem #7 so that the two functions for input & output are replaced with two friends that overload the << & >> operators. See how easy it is to program with this class?

9) Remember separate compilation? Split problem #8 into three files, a header, a main, and a function file. You may need to add namespace(s) to make the whole program run correctly.

10) Starting with the code below get a multi-file Time class program running that demonstrates ALL of the features included in this class. Insure that you actually VERIFY that each feature is working properly. For example: Use the default constructor, and then output to verify that the constructor worked as advertised.
Header1.h
 ifndef HEADER1_H_
 define HEADER1_H_
 include <iostream>

class Time {
 private:
 int hours;
 int minutes;

 public:
 Time();
 Time(int h, int m = 0);
 friend std::ostream & operator<<(std::ostream & os, const Time & t);
};
#endif

Source1.cpp
 include <iostream>
 include "Header1.h"

int main() {
 using std::cout;
 using std::endl;

 // Your code goes here.

 return 0; }

Source2.cpp
 include "Header1.h"

Time::Time() {
 hours = minutes = 0; }

Time::Time(int h, int m) {
 hours = h;
 minutes = m; }

std::ostream & operator<<(std::ostream & os, const Time & t) {
 os << t.hours << " hours, " << t.minutes << " minutes";
 return os; }