CIS 459.22 LAB4

****

Assigned: 5/19/05

Due: 6/1/05 11:59pm

****

Goal

- To use simple operator overloading and friend function

Points

This lab is worth 100 points.

Description

- **Your code will extend the functionality of lab3 with a 2-level menu. You can reuse some fragment of code from lab3.**

- **Your *main()* program should a 2-level menu as shown below.**

  First level main menu:

  1. Date Operations
  2. Time Operations
  3. Quit

  The program should proceed based on the user's choice.

  a) If the user chooses "1" (Date Operations), the program enters the 2nd level menu

  1. Enter two new Date object
  2. Add days to the 1st Date object
  3. Add days to the 2nd Date object
  4. Assign the 2nd Date object to the 1st one

5. Print the 1st and 2nd Date objects
6. Back to main menu

b) If the user chooses "2" (Time Operations), the program enters the 2nd level menu

1. Enter two new Time objects
2. Add hours to the 1st Time object
3. Add hours to the 2nd Time object
4. Assign the 2nd Time object to the 1st one
5. Print the 1st and 2nd Time objects
6. Back to main menu

c) If the user chooses "3" (Quit), exit from the program.

- **To implement the above mentioned menus, we add the following to the Date class:**

1. Add operator = to do assignment
2. Add operator += to add days to a Date object
3. Add operator << to print out a Date object
4. Add operator >> to read in a Date object

The interface of class Date is given as follows.

```
class Date
{
  protected:
      unsigned day;
      unsigned month;
      unsigned year;

      //display an error message if the date format given by the user is
      invalid
      void errmsg(const char* msg);

      //set this object to the specified date
      <return type> set(const char* mmddyy);

  public:
      Date() ;  // default constructor should set the Date object to
      1/1/1950
      Date(const Date &d);  // copy constructor should also print "The
      Date copy constructor is called now"
      Date(const char *mm_dd_yy);  // constructor that initializes with
      the characters input
      // destructor also should print "The Date destructor is called now."
```

```
        Date& operator= (const Date &d);  // performs assignment
        between two Date objects
        Date& operator+= (int days);  // adds days to the current Date
        object
            // Note: 1. We should take care of the fact that  we are only
                    dealing with dates between 01/01/1950 and 12/31/2049.
                    The value of an object cannot go beyond 12/31/2049
            // Note: 2. Be careful about going beyond the month. For
                    example, August 31th plus one day gives September 1st,
                    and the special case with leap year (adding one day to
                    02/28/04 gives us 02/29/04, while adding one day to
                    02/28/03 leads to 03/01/03)
            // Note: 3. Be careful about going beyond the year. For
                    example, 12/31/04 plus one day gives 01/01/05.

        friend istream& operator>> (istream &in, Date &d);
            // reads "mm/dd/yy" from cin and sets the Date object to that
            value if correct
        friend ostream& operator<< (ostream &out, const Date &d);
            // formats and prints the Date object to out as mm/dd/yyyy in
            place of display function
};
```

NOTE: **Please pay attention to the members written in bolds, which identify the extensions from the Date class in lab3.**

- **Similarly, we implement the operator overloading for class Time.**

```
class Time : public Date
{
   protected:
        int hours;
        int minutes;
        int seconds;

        //set this object to the specified time according to the input string
        <return type> set(const char* inputTime);

   public:
        Time();  // default constructor should set the Time object to
        00:00:00 1/1/1950
        Time(const Time &t);  // copy constructor should also print "The
        Time copy constructor is called now."
        Time(const char *inputTime);   // constructor that initializes with
        the characters input
        // destructor also should print "The Time destructor is called now."
```

```
        Time& operator= (const Time &t); // performs the assignment
        between two Time objects
        Time& operator+= (int hours); // adds hours to the time, taking
        care of the validity of the time after addition

        friend istream& operator>> (istream &in, Time &t);
            // reads "hh:mm:ss mm/dd/yy" from cin and sets the time to
            that value if correct
        friend ostream& operator<< (ostream &out, const Time &t);
            // formats and prints the Time object to out as "hh:mm:ss
            mm/dd/yyyy" in place of display function
};
```

NOTE: **Please pay attention to the members written in bolds, which identify the extensions from the Time class in lab3.**

- **You can define any new functions if necessary, as long as the above requirements are fulfilled**

- **Save your program in a modular fashion.**

    - class Date: date.cpp for class implementation of class Date; date.h for class interface definition
    - class Time: time.cpp for class implementation of class Time; time.h for class interface definition
    - lab4.cpp for main function

- **Compile it with *g*++ by entering the following command in the xterm window:**

    ***g*++ -o lab4 lab4.cpp time.cpp date.cpp <return>**

    Remove all compilation errors and warning messages before running the program. Recompile each time you make a change to the program. By finishing, your program should compile and return to the prompt without displaying any error messages.

- **Run your program by entering the following command in the xterm window:**

    lab4 <return>

    Test your program with several cases you may think of, such as "23:30:45 12/31/01", "02/28/00", "12/31/49", "10:30:00 12/12/01". It should comply with the specifications given above.

Submission

- **When you finish with the lab, you need to turn it in for grading. The *submit* command submits your lab electronically. You MUST use the *submit* command to turn in your labs. The format of *submit* command is as follows:**

  ***submit* classname labname files-to-submit**

  where,

  classname is the name of the CIS 459.22 section that you are enrolled in. Your classname is c459.22ab .

  labname is the lab you are working on (lab1, lab2, etc.). For this lab, labname is lab4 .

  files-to-submit is a list of files that make up the lab. For now, it contains lab4.cpp, time.cpp, time.h, date.cpp, and date.h.

  NOTE:

  1. **All of the files in a lab MUST be submitted using one command. If you use two *submit* commands, the second one erases the files from the first submission.**
  2. **Your programs MUST be submitted in source code form. Make sure that you submit the source files only. Do not submit the object file. If you submit the object code, your lab submission will be considered as invalid and late penalty will be imposed.**
  3. **Each *submit* command MUST be entered on one line without pressing Enter. If the line you are entering is too long, it wraps onto the next line.**

Submitting Lab4

To submit your lab for grading, use the following command from the directory which contains all files for this lab:

***submit* c459.22ab lab4 lab4.cpp time.cpp time.h date.cpp date.h**