Data Structure and Algorithm Analysis-COP3530
Program - Unit 4
Total Points: 100

NO LATE ASSIGNMENTS WILL BE ACCEPTED!!

In this assignment will demonstrate your understanding of the following:

1. C++ classes;
2. Implementing a class in C++;
3. Operator overloading with chaining;
4. Preprocessor directives #ifndef, #define, and #endif;
5. this – the pointer to the current object.

In this assignment you will implement the Date class and test its functionality.

Consider the following class declaration for the class date:

```
class Date
{
public:
       Date(); //default constructor; sets m=01, d=01, y      =0001
       Date(unsigned m, unsigned d, unsigned y); //explicit-value constructor to set date equal to today's
              //date. Use 2-digits for month (m) and day (d), and 4-digits for year (y); this function should
              //print a message if a leap year.
       void display();//output Date object to the screen
       int getMonth();//accessor to output the month
       int getDay();//accessor to output the day
       int getYear();//accessor to output the year
       void setMonth(unsigned m);//mutator to change the month
       void setDay(unsigned d);//mutator to change the day
       void setYear(unsigned y);//mutation to change the year

       friend ostream & operator<<(ostream & out, const Date & dateObj);//overloaded operator<< as a friend
function with chaining

    //you make add other functions if necessary
private:
       int myMonth, myDay, myYear; //month, day, and year of a Date obj respectively
};
```

You will implement all the constructors and member functions in the class Date. Please see the comments that follow each function prototype in the Date class declaration above; these comments describe the functionality that the function should provide to the class.

Please store the class declaration in the file "date.h" and the class implementation in the file "date.cpp", and the driver to test the functionality of your class in the file "date_driver.cpp".

Below I have provided a skeleton with stubs and a driver to help you get started. Remember to separate the skeleton into the appropriate files, and to include the appropriate libraries.

Please zip the files date.h, date.cpp and date_driver.cpp in a zip file called “assgn2_pgm_date” and submit electrically through blackboard before the due date and time.

Notes:

1. ALL PROGRAMS SHOULD BE COMPILED USING MS VISUAL STUDIO C++!
2. Information on Month: 1 = January, 2 = February, 3= March, …, 12 = December
3. Test the functionality of your class in “date_driver.cpp” in the following order and include messages for each test:
    a. Test default constructor
    b. Test display
    c. Test getMonth
    d. Test getDay
    e. Test getYear
    f. Test setMonth
    g. Test setDay
    h. Test setYear
4. See sample output below.
5. See skeleton below.

S A M P L E   O U T P U T   FOR Assignment#2

```
Default constructor has been called
01/01/0001

Explicit-value constructor has been called
12/31/1957

Explicit-value constructor has been called
Month = 15 is incorrect

Explicit-value constructor has been called
2/29/1956
This is a leap year

Explicit-value constructor has been called
Day = 30 is incorrect

Explicit-value constructor has been called
Year = 0000 is incorrect

Explicit-value constructor has been called
Month = 80 is incorrect
Day = 40 is incorrect
Year = 0000 is incorrect

12/31/1957

12
31
1957
myDate: 11/12/2015  test2Date:  02/29/1956  yourDate: 12/31/1957
```

Skeleton FOR Assignment#2

```
#include <iostream>
#include <string>
//#include "date.h"

using namespace std;



//*************************************************************************************
//*************************************************************************************
//      D A T E . h

//This declaration should go in date.h

#ifndef DATE_H
#define DATE_H

class Date
{
public:
       Date(); //default constructor; sets m=01, d=01, y      =0001
       Date(unsigned m, unsigned d, unsigned y); //explicit-value constructor to set date equal to today's
              //date. Use 2-digits for month (m) and day (d), and 4-digits for year (y); this function should
              //print a message if a leap year.
       void display();//output Date object to the screen
       int getMonth();//accessor to output the month
       int getDay();//accessor to output the day
       int getYear();//accessor to output the year
       void setMonth(unsigned m);//mutator to change the month
       void setDay(unsigned d);//mutator to change the day
       void setYear(unsigned y);//mutation to change the year

       friend ostream & operator<<(ostream & out, const Date & dateObj);//overloaded operator<< as a friend
function with chaining

    //you make add other functions if necessary
private:
       int myMonth, myDay, myYear; //month, day, and year of a Date obj respectively
};

#endif


//*************************************************************************************
//*************************************************************************************
//      D A T E  . C P P


//This stub (for now) should be implemented in date.cpp
//*****************************************************************************
//Name:  Date
//Precondition: The state of the object (private data) has not been initialized
//Postcondition: The state has been initialized to today's date
//Description:  This is the default constructor which will be called automatically when
//an object is declared.  It will initialize the state of the class
//
//*****************************************************************************
Date::Date()
{


       //the code for the default constructor goes here


}
```

```
//*****************************************************************************************
//Name:  Date
//Precondition:
//Postcondition:
//Description:
//
//
//*****************************************************************************************
Date::Date(unsigned m, unsigned d, unsigned y)
{



}

//*****************************************************************************************
//Name:  Display
//Precondition:
//Postcondition:
//Description:
//
//
//*****************************************************************************************
void Date::display()
{



}

//*****************************************************************************************
//Name:  getMonth
//Precondition:
//Postcondition:
//Description:
//
//
//*****************************************************************************************
int Date::getMonth()
{

       return 1;

}

//*****************************************************************************************
//Name:  getDay
//Precondition:
//Postcondition:
//Description:
//
//
//*****************************************************************************************
int Date::getDay()
{

       return 1;

}

//*****************************************************************************************
//Name:  getYear
//Precondition:
//Postcondition:
//Description:
```

```
//
//
//*************************************************************************************
int Date::getYear()
{

       return 1;

}

//*************************************************************************************
//Name: setMonth
//Precondition:
//Postcondition:
//Description:
//
//
//*************************************************************************************
void Date::setMonth(unsigned m)
{



}

//*************************************************************************************
//Name:  setDay
//Precondition:
//Postcondition:
//Description:
//
//
//*************************************************************************************
void Date::setDay(unsigned d)
{




}

//*************************************************************************************
//Name:  getYear
//Precondition:
//Postcondition:
//Description:
//
//
//*************************************************************************************
void Date::setYear(unsigned y)
{




}

ostream & operator<<(ostream & out, const Date & dateObj)
{



return out;

}
```

```
//*********************************************************************************************
//*********************************************************************************************
//     D A T E  D R I V E R . C P P




//EXAMPLE OF PROGRAM HEADER
/********************************************************************************************

Name:                                 Z#:
Course: Date Structures and Algorithm Analysis (COP3530)
Professor: Dr. Lofton Bullard
Due Date:                      Due Time:
Total Points: 100
Assignment 3: Date program

Description:



*********************************************************************************************/


int main()
{

   //Date myDate;
   //Date yourDate(12,31, 1957);
   //Date test1Date(15, 1, 1962); //should generate error message that bad month
   //Date test2Date(2, 29, 1956); //ok, should say leep year
   //Date test3Date(2, 30, 1956); //should generate error message that bad day
   //Date test4Date(12,31,0000); //should generate error message that bad year
   //Date test5Date(80,40,0000); //should generate error message that bad month, day and year

   //yourDate.display();
   //cout<<yourDate.getMonth()<<endl;
   //cout<<yourDate.getDay()<<endl;
   //cout<<yourDate.getYear()<<endl;
   //myDate.setMonth(11);
   //myDate.setDay(12);
   //myDate.setYear(2015);
   //cout<<"myDate: "<<myDate<<" test2Date: "<<test2Date<<" yourDate: "<<yourDate<<endl;

   return 0;

}
```