CS 271
Programming Assignment # 8

You are expected to read Chapter 16 before beginning this assignment. Also, view the video by Derek Banas on C++ Classes.

C++ style input and output is required throughout this lab.

The following C language features are prohibited:

- printf
- scanf
- C library names
- \t in output statements
- \n in output statements

There are quite a few files for this assignment. Zip them up and submit a zip file.

makefile
Time.h
Time.cpp
Date.h
Date.cpp
Calendar.cpp (this is the driver program)

Zip all 6 files together and submit one .zip file.

Yes, you must submit Time.h even though the code was given to you.

Programs must compile.
Programs that have syntax errors receive a grade of zero.

- Documentation and Style - This is a large part of the grade on this assignment. See the "Documentation and Style Guidelines" document in the Lecture Notes Module. The TA will be checking for:
  - Header and inline comments
  - Indentation
  - Spacing
  - Use of blank lines
- Follow these conventions for identifiers:
  - function names - start with lowercase letter, then use camel case
  - variable names and data member names - start with lowercase letter, then use camel case
  - class names - start with uppercase letter, then use camel case
  - constants - all uppercase letters
- Follow convention for naming accessors and mutators.
  - mutator - starts with lowercase "set" followed by, first letter capitalized, name of the data member
  - accessor - starts with lowercase "get" followed by, first letter capitalized, name of the data member

- Programs must declare ALL variables at the top of the function. C++: The only exception is a variable that is used to control a for loop.

```
int main ( void ) {

   // all variable declarations first

   // then executable statements
}
```

- Header files must have a preprocessor wrapper. We did this in C and we'll continue doing this in C++. The preprocessor constant name must be all caps and contain the file name, an underscore, and H. Example:

```
#ifndef TIME_H
#define TIME_H
.
.
.
#endif
```

- Do not use the word "this" in your code.

  In C++, when we use the keyword "this" inside a function, it is a pointer to the calling object. For now, that may not make any sense to you.

  There will be times when we have to use "this". Lab 8 is <u>not</u> the time. Don't use "this" in your code.

- Do not use "std::" in your code.

  By placing the statement

```
using namespace std;
```

  at the top of your programs, you avoid the need to put std:: in front of all the standard library identifiers. This is explained in chapter 15 (section 15.3). If you look at the very first program in chapter 15, you can see how many times you would have to put "std::" in this short program. You must put "using namespace std;" in your programs for this class.

```
#include <iostream>

int main( )
{
   std::cout << "Enter first integer: ";
   int number1;
   std::cin >> number1;
   std::cout << "Enter second integer ";
   int number2;
   std::cin >> number2;
   int sum = number1 + number2;
   std::cout << "Sum is " << sum << std::endl;
}
```

Creating constants and symbolic constants

- Creating a symbolic constant.

```
#include <stdio.h>
#define SIZE 10          // SIZE is now a symbolic constant
                         // Any place where SIZE appears in the program,
                         // it will be replaced by 10.  This is a text
                         // based "search and replace" and is done by the
                         // preprocessor...BEFORE the program is sent to
                         // the compiler.
```

- Declaring a local constant using "const".

```
void checkHour(int x) {

   const int HOUR_LIMIT = 12;          // The constant HOUR_LIMIT is
                                       // only in scope inside the
                                       // checkHour function.

   // executable statements begin here

   if (x <= HOUR_LIMIT)
       cout << "x is a valid hour" << endl;
}
```

Random Number Generation

We did this in C. Below is the C++ version of random number generation. The function names and the way you use them are the same, but the library names are different.

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main ( ) {

   // generate a random number between 1 and 10

   const int MAX = 10, num;
   srand(time(NULL));
   num = rand() % MAX + 1;
   cout << "num is " << num << endl;

}
```

Makefile

A makefile is required for this assignment.

The makefile for C++ programs is very similar to those you created for C. Instead of using gcc, you use

```
g++ -std=c++11
```

You must have separate targets for

- all (produces executable named "Calendar")
- Calendar
- Calendar.o
- Date.o
- Time.o.

UML Class Diagram for Class Time

Time
- hour : int - minute : int - second : int
<<constructor>> Time() <<constructor>> Time(h : int, m : int, s : int) + setHour(h : int) + setMinute(m : int) + setSecond(s : int) + getHour( ) : int + getMinute( ) : int + getSecond( ) : int + print( ) + print12( )

UML stands for "Unified Modeling Language". UML is a set of diagramming structures for representing various aspects of computer systems. We will be using only one UML diagram type: the class diagram.

There are 3 boxes in a class diagram.

1) top box: Contains the class name in bold font, centered.
2) middle box: Contains the data elements (also called attributes or properties). In C++ the specific term used is "data members".
3) bottom box: Contains the behaviors or actions. In C++ the specific term used is "member functions".

As a group, the data members and member functions of a class are referred to as the "members of the class".

There is a very particular way to write the data members and member functions in a UML diagram. Look at the UML diagram above for the Time class.

Accessibility Modifiers

- (minus sign) indicates the member is private, accessible only to other members of the same class
+ (plus sign) indicates the member is public, accessible to any program
# (hashtag) indicates the member is protected (we will not use this in 271)

The accessibility modifier may be omitted on constructors. In that case it is assumed to be public.

<< constructor >> This special notation is placed in front of constructors.

Parameters

In UML the parameters are written with the name first, then a colon, then the type.

Return Type

In UML a colon and the return type of a function are written after the parameter list. The return type may be omitted if the function's return type is void.

Time.h

This file is given. Copy it from my public_html folder.

```
cp /home/faculty2/esteiner/public_html/cs271/Time.h .
```

Reminder: space period at the end of the copy command means copy the file to the current folder.

The Time class represents a 24-hour time.

The header file Time.h contains the class definition. Open the file and inspect the code. The class definition begins on line 7.

The data members are declared in the **private** section of the class definition and the member function prototypes are placed in the **public** section.

For those of you who know Java... class definitions are a language feature where C++ is EXTREMELY DIFFERENT from Java.

The data members are:

- hour (unsigned int)
- minute (unsigned int)
- second (unsigned int)

"unsigned int" is used when you do not want to allow negative numbers to be stored in the variable.

*** In the file Time.h, locate the declaration of the data members.

Yes, there is a Time class in the textbook. It may give you some ideas. But, it doesn't meet the requirements for this assignment.

The member functions are:

- default constructor Time( ) - initializes Time object to 00:00:00
- constructor Time(int h, int m, int s) - initializes Time object using the parameters. In this function we will call the mutators, one-by-one, to set values of the data members.
- 3 mutators: a mutator for hour, a mutator for minute, and a mutator for second. (In this class we will take the "ignore it approach" for invalid data. We will not change the data member, not print any error messages, and not throw an exception.)
- 3 accessors: an accessor for hour, an accessor for minute, and an accessor for second.
- print ( ) - prints the time in 24-hour format `hh:mm:ss` (two digits each). In the function we will call the accessors to get the values of the data members.
- print12 ( ) - prints the time in 12-hour format `hh:mm:ss` (two digits each) followed by one space, then AM or PM. In this function we will call the accessors to get the values of the data members.

*** In the file Time.h, locate the prototype for each function. There are 10 functions total.

At the end of the class definition, there is a curly brace **AND A SEMICOLON.**

Every class definition must end with `};`

Time.cpp

Make a copy of Time.h and save it in a new file called Time.cpp.

Edit the Time.cpp file. Remove the following lines:

- class Time {
- public:
- private:
- };

Insert the following statement before "using namespace std;"

```
#include "Time.h"
```

One function at a time:

- Add `Time::` in front of each function name. This tells the compiler that the function is a member of the Time class. (It goes after the return type, before the function name.)

- Remove the semicolon at the end of the function prototype. Replace it by { and then write the function definition. Close the function with }

After you write each function, stop, save, and compile:

```
g++ -std=c++11 -c Time.cpp
```

Debug any syntax errors.

Example: Here is what the function definition should look like for the mutator for hour:

```
void Time::setHour( int h ) {

   if (h >= 0 && h <= 23)              // See note on the next page
       hour = h;                       // about "this".

   // There is no "else".  We are ignoring invalid data.
}
```

Continue until you have written all 10 functions and the code compiles without error.

Regarding the use of "this" inside member functions.

Do not use "this" unless it's absolutely necessary.

That means DO NOT USE "this" in any of the member functions in this assignment.

If you don't know what "this" is in C++, that's fine for now. You won't need it for this assignment.

Date.h

Create a file named Date.h. (If you want to copy Time.h, you can do that. Just make sure that you get all of the function names, parameter names, etc. changed.) Include the preprocessor wrapper, #include the needed C++ libraries, and write the "using" statement.

Yes, there is a Date class in the textbook. It may give you some ideas. But, it doesn't meet the requirements for this assignment.

Write the class definition for class Date. This class represents a typical date with month, day and year. Values may range from 1/1/1980 to 12/31/2100.

Remember that the class definition contains only data members and member function prototypes.

The data members are:

1) month (unsigned int)
2) day (unsigned int)
3) year (unsigned int)

By convention, data members are private.

The member functions are:

1) default constructor Date( ) - initializes Date object to 1/1/1980
2) another constructor Date(int m, int d, int y) - initializes Date object using the parameters. This constructor **must** use the mutators to set values of the data members.
3) 3 mutators: a mutator for month, a mutator for day, and a mutator for year. (Use the "ignore it approach" for invalid data just like we did in the Time class.)
4) 3 accessors: an accessor for month, an accessor for day, and an accessor for year.
5) print ( ) - prints the Date in the format `mm/dd/yyyy` (two digits each for month and day, four digits for year). Use the accessors to get the values of the data members.

Date.cpp

Copy Date.h to a new file called Date.cpp. Use the same process as you did for the Time class to remove unneeded lines and convert the prototypes to full function definitions.

Write the function definitions for the 9 member functions of the Date class.

Remember that you cannot have any magic numbers in the executable portion of the code.

Incremental Programming : Compile Date.cpp before you go on to the next step.

Calendar.cpp

Create a driver program with a main function. In the main function, create an array of 5 Time objects and set them to random times. (Specifically, for each element in the array, generate a random integer and call setHour; generate a random integer and call setMinute; generate a random integer and call setSecond.)

Print the array, one element per line, with 24 hour format. Include meaningful text in your output.

Print the array again, one element per line, with 12 hour format. Include meaningful text in your output.

Create an array of 5 Date objects. Use a loop to input the month, day and year for each Date. Then call the mutators to set those data members.

Print the elements of the array, one element per line. Include meaningful text in your output.

Add calls to the other member functions as needed to test them thoroughly.

There are quite a few files for this assignment. Zip them up and submit a zip file.

1. makefile
2. Time.h
3. Time.cpp
4. Date.h
5. Date.cpp
6. Calendar.cpp (this is the driver program)

Zip all 6 files together and submit one .zip file.

Yes, you must submit Time.h even though the code was given to you.

For zip software, on Windows I use 7Zip.
On Mac you might try iZip.