Lab 6

Time class
Date class

There are versions of these in the textbook. They are not the same as what is needed for the lab assignment.

Section 17.2

```
#ifndef TIME_H
#define TIME_H

class Time {

   public:
         Time( );
         void setTime( int, int, int );
         void printUniversal( );
         void printStandard( );

   private:
        int hour;
        int minute;
        int second;

};
#endif
```

This would be stored in Time.h

Function Implementations (Function Definitions)

This would be in the file Time.cpp

```
#include <iostream>
#include <iomanip>
#include <stdexcept>

#include "Time.h"
using namespace std;

Time::Time( ) {

   hour = minute = second = 0;
}

void Time::setTime( int h, int m, int s ) {

   // validate the data
   if ( ( h >= 0 && h < 24 )  && ( m >= 0 && m < 60 )
         && ( s >=0 && s < 60 ) )

   {
        hour = h;
        minute = m;
        second = s;
   }
   else
      throw invalid_argument( "hour, minute and/or second was out of range");
} // end function setTime
```

```
void Time::printUniversal( )
{
    cout << setfill( '0' ) << setw( 2 ) << hour << ":" << setw( 2 ) << minute
             << ":" << setw ( 2 ) << second;

    cout << setfill( ' ' );

}

void Time::printStandard( ) {

      cout << setfill( '0' ) << ( ( hour == 0 || hour == 12 ) ? 12 : hour % 12 )
               << ":" << setw( 2 ) << minute << ":" << setw ( 2 ) << second <<
               ( hour < 12 ? " AM" : " PM" );

      cout << setfill ( ' ' );

}
```

Program to test the Time class

this will be stored in a file named TimeTest.cpp

```
#include <iostream>
#include <iomanip>
#include "Time.h"
using namespace std;

int main ( ) {

    Time t;

    cout << "The universal time is ";
    t.printUniversal( );

    cout << "The standard time is ";
    t.printStandard( );

    t.setTime( 13, 27, 6 );

    // print universal and standard time again

   try {

        t.setTime( 99, 99, 99 );
    }
    catch ( invalid_argument &e )  {

           cout << "Exception:  " << e.what( ) << endl << endl;
    } // end catch

    // print universal and standard time again
    // to verify that the values didn't change

} // end main
```

& in the parameters

C++ has something called reference parameters. It's like pointers but without the extra punctuation.

```
void myfunction (int n)  // value parameter

void myfunction ( int &n ) {  // reference parameter

    // n is a reference parameter
    // in C++ the wording is " another name for "

    n = 10 * n + 4;

}
```

The variable in calling function is changed.

```
int main( ) {

     int k = 13;
     myfunction ( k );
     cout << "k is " << k << endl;
}
```

& can also be used as address of
&& in the logical "and" operator
& as a bitwise "and" operator

* multiplication
* dereferencing a pointer
* declaring a pointer

Page 633 extended example using reference variables &, pointers *, and primitive variable.

Date class

Data members: month, day, and year

Member functions: constructor, accessors, mutators, print( ), etc.

Use cin for input and cout for output.