```
//--------------------------------------------------------------------
//
//  Laboratory 1                                           logbook.h
//
//  Class declaration for the Logbook ADT
//
//--------------------------------------------------------------------
class Logbook
{
  public:

    // Constructor
    Logbook ( int month, int year );        // Create a logbook

    // Logbook marking operations
    void putEntry ( int day, int value );   // Store entry for day
    int getEntry ( int day ) const;         // Return entry for day

    // General operations
    int month () const;                     // Return the month
    int year () const;                      // Return the year
    int daysInMonth () const;               // Number of days in month

    // In-lab operations
    void displayCalendar () const;          // Display as calendar
    Logbook ();                             // Default constructor
    void putEntry ( int value );            // Store entry for today
    int operator [] ( int day ) const;      // Return entry for day
    void operator += ( const Logbook &rightLogbook );
                                            // Combine logbooks
  private:

    // Facilitator (helper) function
    int leapYear () const;                  // Leap year?

    // In-lab facilitator function
    int dayOfWeek ( int day ) const;        // Return day of the week

    // Data members
    int logMonth,     // Month covered by logbook
        logYear,
        entry [31];   // Logbook entries
};
```

```
//--------------------------------------------------------------------
//
//  Laboratory 1                                          logbook.cpp
//
//  SOLUTION: Implementation of the Logbook ADT
//
//--------------------------------------------------------------------

#include <iostream.h>
#include <iomanip.h>
//#include <dos.h>
#include "logbook.h"

//--------------------------------------------------------------------

Logbook::Logbook ( int month, int year )

// Constructor. Creates an empty logbook for the specified month.

  : logMonth(month),
    logYear(year)
{
    int j;   // Loop counter

    for ( j = 0 ; j < 31 ; j++ )
        entry[j] = 0;
}

//--------------------------------------------------------------------

void Logbook::putEntry ( int day, int value )

// Stores entry for the specified day.

{
    entry[day-1] = value;
}

//--------------------------------------------------------------------

int Logbook::getEntry ( int day ) const

// Returns entry for the specified day.
```

```
{
    return entry[day-1];
}

//--------------------------------------------------------------------

int Logbook::month () const

// Returns the logbook month.

{
   return logMonth;
}

//--------------------------------------------------------------------

int Logbook::year () const

// Returns the logbook year.

{
   return logYear;
}

//---------------------------------------------------------------------

int Logbook::daysInMonth () const

// Returns the number of days in the logbook month.

{
    int result;   // Result returned

    switch ( logMonth )
    {
      case 4 : case 6 : case 9 : case 11 :
           result = 30;
           break;
      case 2 :
           if ( leapYear() )
              result = 29;
           else
```

```
              result = 28;
           break;
      default :
           result = 31;
    }

    return result;
}

//--------------------------------------------------------------------

int Logbook::leapYear () const

// If the logbook month occurs during a leap year, then returns 1.
// Otherwise, returns 0.

{
    return ( logYear % 4 == 0   &&
             ( logYear % 100 != 0  ||  logYear % 400 == 0 )  );
}


//--------------------------------------------------------------------
//
//  Laboratory 1                                          test1.cpp
//
//  Test program for the operations in the Logbook ADT
//
//--------------------------------------------------------------------

#include <iostream.h>
#include "logbook.h"

void main()
{
    int month,   // Input month
        year,    // Input year
        day,     // Input day
        entry,   // Input logbook entry
        dofw,    // Day of the week
        stop,    // Signals end of test
        j;       // Loop counter

    // Create a logbook (not used in Test 4).
    cout << endl << endl
         << "Enter the month and year for the logbook month : ";
    cin >> month >> year;
    Logbook testLog(month,year);

    // Test 1 : Tests the month, year, and daysInMonth operations.

    cout << "Month : " << testLog.month() << endl;
    cout << "Year  : " << testLog.year() << endl;
    cout << "# days in month : " << testLog.daysInMonth() << endl;

    // Test 2 : Tests the putEntry and getEntry operations.

    stop = 0;
    while ( !stop )
    {
        cout << endl << "Enter day and entry (0 0 to exit Test 2) : ";
        cin >> day >> entry;
        if ( day != 0  &&  entry != 0 )
        {
           testLog.putEntry(day,entry);
           cout << "Logbook:" << endl;
           for ( day = 1 ; day <= testLog.daysInMonth() ; day++ )
           {
               cout << day << " " << testLog.getEntry(day) << '₩t';
               if ( day % 5 == 0 )
                  cout << endl;
           }
           cout << endl;
        }
        else stop = 1;
    }

     // Test 3 : Tests the calendar operation.

//3  cout << endl;
//3  testLog.displayCalendar();
//3  cout << endl;

//4  // Test 4 : Tests the overloaded constructor and putEntry
//4  //          operations.
//4
```

```
//4  Logbook thisMonth;
//4  cout << endl << "Logbook for this month:" << endl;
//4  cout << "Month : " << thisMonth.month() << endl;
//4  cout << "Year  : " << thisMonth.year() << endl;
//4  cout << "# days in month : " << thisMonth.daysInMonth() << endl;
//4  thisMonth.putEntry(100);
//4  for ( day = 1 ; day <= thisMonth.daysInMonth() ; day++ )
//4  {
//4      cout << day << " " << thisMonth.getEntry(day) << '₩t';
//4      if ( day % 5 == 0 )
//4         cout << endl;
//4  }
//4  cout << endl;

//5  // Test 5 : Tests the [] operation.
//5
//5  cout << "Logbook:" << endl;
//5  for ( day = 1 ; day <= testLog.daysInMonth() ; day++ )
//5  {
//5      cout << day << " " << testLog[day] << '₩t';
//5      if ( day % 5 == 0 )
//5         cout << endl;
//5  }
//5  cout << endl;

//6  // Test 6 : Tests the += operation.
//6
//6  Logbook logDay100(month,year),
//6          logDay200(month,year);
//6
//6  cout << endl
//6       << "Loading logbooks logDay100 and logDay200" << endl;
//6  for ( day = 1 ; day <= logDay100.daysInMonth() ; day++ )
//6  {
//6     logDay100.putEntry(day,100*day);
//6     logDay200.putEntry(day,200*day);
//6  }
//6
//6  logDay100 += logDay200;
//6
//6  cout << "Combined logbooks:" << endl;
//6  for ( day = 1 ; day <= logDay100.daysInMonth() ; day++ )
//6  {
```

```
//6      cout << day << " " << logDay100.getEntry(day) << '₩t';
//6      if ( day % 5 == 0 )
//6         cout << endl;
//6  }
//6  cout << endl;
}
```