**Homework No. 3**

**Determining Leap Years**

**Objective**

This third assignment involves the use of selection structures, logical expressions, and user-defined functions. As in the previous homework, a C++ program is required to be developed from scratch, debugged, and tested. Leap years have 366 days, while ordinary years have 365. The required program will determine if a user-supplied year is: (1) a valid year; and (2) whether it is a leap year or not. Appropriate messages are then produced depending on the condition detected about the input year.

**Problem Statement and Analysis**

As the homework page stated, we need a program that takes a year (after 1752) as input and outputs the number of days in the year after determining whether or not the input year represents a leap year. If the input is a year prior to 1752, the program outputs a message explaining why it does not accept such input. The program will not compute an answer for such a value. The sample input given in the homework description is 2000. For this, the program will produce the output:

The year 2000 is a leap year, and has 366 days.

If the input were 500, the program produces the output:

The year 500 is not a valid input year.

**Algorithm**

We see the need for an input variable:

```cpp
int inYear  // the input year (an int variable)
```

We are instructed to create a function whose prototype is as follows:

```cpp
bool isLeapYear( int year );
// determines whether input year is leap year or not
```

To help in the design and development of the function, the website given in the homework description indicates that a year (after 1752) is a leap year if the following conditions hold:

- The year is divisible by four but not by 100 (i.e., not a century year); or
- If the year is in fact a century year, it is also divisible by 400.
If these conditions fail to hold, then the year is a regular year (i.e., not a leap year).

The algorithm is as follows:

1. Print a suitable prompt for the user;
2. Accept the input year inYear;
3. If inYear is not greater than 1752, then print an appropriate message indicating the inYear is not a valid year;
4. Otherwise, test inYear by invoking the function isLeap(inYear) to determine if inYear is a leap year or not.
5. Print a suitable message depending on the returned value of isLeap(inYear).

The algorithm for the function isLeap(inYear) is as follows:

1. If inYear is not divisible by 4, then we return false.
2. Otherwise (i.e., inYear is divisible by 4), if inYear is not divisible by 100, then we return true.
3. Otherwise (i.e., inYear is divisible by 100), if inYear is not divisible by 400, then we return false.
4. Otherwise (i.e., inYear is divisible by 400), return true.
5. Print a suitable message depending on the value of isLeap(inYear).

**Program Implementation**

The program is given below in text format. A screen capture of the compilation is then supplied immediately after:

```cpp
#include <iostream>
using namespace std;

//function prototype
bool isLeap(int year); //is year a leap year?
```
```cpp
int main() {
    int inYear = 0;

    // Accept the input year
    cout << "Enter a year greater than 1752: ";
    cin >> inYear;

    // Determine if input year is valid
    if (inYear < 1752)
        cout << "The year " << inYear << " is not a valid year." << endl;
    else
        // Test if input year is a leap year and then
        // print appropriate messages depending on test
        if (isLeap(inYear))
            cout << "The year " << inYear << " is a leap year, and has 366 days." << endl;
        else
            cout << "The year " << inYear << " is not a leap year, and has 365 days." << endl;

    system("PAUSE");
    return 0;
}

// Function implementation
bool isLeap(int year) { // is year a leap year?
    // Determine if input year is divisible by 4
    if (year % 4 != 0)
        return false;
    else
        // Determine if input year is divisible by 100
        if (year % 100 != 0)
            return true;
        else
            // Determine if input year is divisible by 400
            if (year % 400 != 0)
                return false;
            else
                return true;
} // end isLeap
```
Program Testing and Verification

The test suite and the obtained values from several runs are tabulated below:

<table>
<thead>
<tr>
<th>Input Year</th>
<th>Leap Year?</th>
<th>Program Output</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>Not valid</td>
<td>“Not valid”</td>
<td>As expected</td>
</tr>
<tr>
<td>2004</td>
<td>Yes</td>
<td>“Leap Year”</td>
<td>As expected</td>
</tr>
<tr>
<td>1900</td>
<td>No</td>
<td>“Not Leap Year”</td>
<td>As expected</td>
</tr>
<tr>
<td>2005</td>
<td>No</td>
<td>“Not Leap Year”</td>
<td>As expected</td>
</tr>
<tr>
<td>1854</td>
<td>No</td>
<td>“Not Leap Year”</td>
<td>As expected</td>
</tr>
<tr>
<td>2500</td>
<td>No</td>
<td>“Not Leap Year”</td>
<td>As expected</td>
</tr>
</tbody>
</table>

Screen capture of the first test run:
Screen capture of the second test run:

```
Enter a year greater than 1752: 2004
The year 2004 is a leap year, and has 366 days.
Press any key to continue . . .
```

Screen capture of the third test run:

```
Enter a year greater than 1752: 1900
The year 1900 is not a leap year, and has 365 days.
Press any key to continue . . .
```

Screen capture of the fourth test run:

```
Enter a year greater than 1752: 2005
The year 2005 is not a leap year, and has 365 days.
Press any key to continue . . .
```

Screen capture of the fifth test run:

```
Enter a year greater than 1752: 1854
The year 1854 is not a leap year, and has 365 days.
Press any key to continue . . .
```

Screen capture of the sixth test run:

```
Enter a year greater than 1752: 2500
The year 2500 is not a leap year, and has 365 days.
Press any key to continue . . .
```
As can be seen, all results were as expected signaling a high probability that the program is correct.

**Conclusion**

In this assignment, we analyzed a problem, set up an algorithm based on the analysis of the definition of a leap year, implemented the algorithm in a C++ program, entered the code using a text editor, compiled it, corrected the syntax errors, linked the object code, ran the resulting executable, got rid of the logic errors revealed by the runs, corrected the logic errors, recompiled, relinked, ran the modified executable, and repeated this cycle until further test runs indicated that the program was most probably correct according to the specification.

The results of the test runs were examined and compared with the expected outcome. All output was as expected.

All in all, an interesting exercise and an informative one (learning that certain century years will not be leap years, etc.).