University of Massachusetts Dartmouth
Department of Electrical and Computer Engineering

ECE 160 Name: `dayofweek.cpp`
Project 6 Due: See `http://ece160.org`

Write a program that, given a date (month, day, year) will display the corresponding day of the week. The program should then ask for another date, and repeat the process until a year of zero is entered.

Calculating a day for any given date is somewhat complicated, as Emperor Augustus made changes to the calendar in 8AD, and Pope Gregory XIII eliminated 11 days in 1582 (adopted by Great Britain and America in 1752...Wednesday September 2nd was followed by Thursday September 14th). The algorithm discussed in this project will only work from 1753 onward.

The overall algorithm, given a *mm*, *dd*, and *yyyy*, is to find the number of days between 01/01/0001 and *mm*/*dd*/*yyyy* (accounting for lost days mentioned above). This number mod 7 gives a value between 0 and 6, where 0 represents Sunday, 1 represents Monday, and so on, with 6 representing Saturday.

The first step is to find the number of days (accounting for the changes to the calendar above) between 01/01/0001 and 12/31/(yyyy-1). The formula is:

$$numdays = (y-1)*365 + \frac{y-1}{4} - \frac{y-1}{100} + \frac{y-1}{400}$$

The next step is to add the number of days in all of the full months in the current year. For example if *mm* is 5, you would add 31 + 28 + 31 + 30 to *numdays*. For this part, assume February always has 28 days.

Next, add *dd* to *numdays*. This is [almost] the total number of days between 01/01/0001 and *mm*/*dd*/*yyyy* entered.

There is one last refinement needed. If the current year is a leap year, and the desired date is on or after March 1, you need to add 1 to *numdays*. A year is a leap year if the year is evenly divisibly by four, unless the year is evenly divisible by 100, in which case it is not a leap year, unless the year is evenly divisible by 400, in which case it is a leap year (yes, it sound crazy, but this is the way it is). There is, in fact, another correction to be made at 2900 year intervals, but we'll ignore that one. The following logical expression may be used to determine if the year is a leap year.

```
((year%4==0) && (year%100!=0)) || (year % 400 ==0)
```

OR

```
(!(year % 4) && (year % 100) ) || !(year % 400)
```

The remainder (remember %) after dividing *numdays* by 7 gives you a *daycode*. The day is determined by the *daycode* as follows:

Day 0: Sunday	Day 2: Tuesday	Day 4: Thursday	Day 6: Saturday
Day 1: Monday	Day 3: Wednesday	Day 5: Friday	

Initially test your program with the following dates:

February 28, 1900	March 1, 1900	February 28, 1955
March 1, 1955	February 28, 1996	March 1, 1996
February 28, 2000	March 1, 2000	January 1, 1997
December 31, 1997	September 24, 2007	September 26, 2007

In addition to the values above and below, you should test your program for several values (especially around Feb 28/29/Mar 1, and Dec 31/Jan 1). A handy resource may be found at `http://www.timeanddate.com/calendar/`

A sample run of the program might look as follows (user input underlined):

```
Date (m d y): 2 28 1900
 2/28/1900 => Wednesday

Date (m d y): 3 1 1900
 3/ 1/1900 => Thursday

Date (m d y): 2 28 1955
 2/28/1955 => Monday

Date (m d y): 3 1 1955
 3/ 1/1955 => Tuesday

Date (m d y): 2 28 1996
 2/28/1996 => Wednesday

Date (m d y): 3 1 1996
 3/ 1/1996 => Friday

Date (m d y): 2 28 2000
 2/28/2000 => Monday

Date (m d y): 3 1 2000
 3/ 1/2000 => Wednesday

Date (m d y): 1 1 1997
 1/ 1/1997 => Wednesday

Date (m d y): 12 31 1997
12/31/1997 => Wednesday

Date (m d y): 9 23 2013
 9/23/2013 => Monday

Date (m d y): 9 25 2013
 9/25/2013 => Wednesday

Date (m d y): 0 0 0
Press any key to continue . . .
```

Remember to include the required certification found at the end of the Grading Rubric. Also, check the rubric before handing in your project to insure you haven't made any obvious errors that result in loss of points. Remember the cases above are not an exhaustive list. I will try several other values...you should too.

Hints and recommendations:

It is strongly suggested that you create (at least) two functions which will make the calculations easier. Whereas you do not have to create these functions for this program, you will need to use them in the next program. Whether you write them now or later is up to you.

The first function should have a prototype:

```
int getndim(int m, int y);
```

This function should return the number of days in the specified month (`m`) of the specified year (`y`).

The second function should have a prototype:

```
int getdaycode(int m, int d, int y);
```

This function should calculate the `daycode` value discussed on page 1. This function may (should) make use of the `getndim()` function.

A third function to consider is a function with the prototype:

```
void printdayname(int day);
```

This function will simply print the name of a given day. If `day` is `0`, then `Sunday` is printed; `1` prints `Monday`, and so on…`6` prints `Saturday`.