Write a program that reads a sequence of dates, and for each one prints the corresponding weekday, or prints that the date is not correct according to the Gregorian calendar.

Implement the functions

```c
bool is_leap_year (int year);
bool is_correct_date (int day, int month, int year);
string weekday(int day, int month, int year);
```

Given a year \( y \), `is_leap_year (y)` tells if \( y \) is a leap year or not. Given a date defined with \( d, m \) and \( y \), `is_correct_date (d, m, y)` tells if the date is correct or not according to the Gregorian calendar. Given a correct date defined with \( d, m \) and \( y \), `weekday(d, m, y)` returns the corresponding weekday (that is, “Monday”, “Tuesday”, ...).

To know the weekday, use the congruence of Zeller: Given a date defined by the triple \((d, m, y)\), where \( d \) is the day, \( m \) is the month, and \( y \) is the year,

1. Subtract two units to the month \( m \), and if the result is zero or less, add 12 to the month and subtract a unit to the year. Call \( m' \) the new month and call \( y' \) the new year.
2. Compute the century \( c \) (the first two digits of the year) from the year \( y' \).
3. Compute the year inside the century \( a \) (the two last digits of the year) from the year \( y' \).
4. Compute
   \[
   f = \lfloor 2.6m' - 0.2 \rfloor + d + a + \lfloor a/4 \rfloor + \lfloor c/4 \rfloor - 2c.
   \]
5. Finally, \( f \) modulo 7 give us the desired result: 0 represents Sunday, 1 represents Monday, 2 represents Tuesday...

You can find the rule about leap years in the exercise `LINK::problem://problemsjutge.org:problem`.

**Precondition**

For the functions `is_leap_year ()` and `is_correct_date ()`, the value of the year is always between 1800 and 9999 (both included). For the function `weekday()`, the given date is always correct w.r.t. the function `is_correct_date ()`.

**Input**

Each date of the input is composed by three integers, corresponding respectively to the day, the month and the year. All the years are between 1800 and 9999.

**Output**

For each date of the input, print in a line the corresponding weekday (“Monday”, “Tuesday”, “Wednesday”, “Thursday”, “Friday”, “Saturday”, “Sunday”) if it is a correct date according to the Gregorian calendar, or “Incorrect Date” if it is not.
Sample input
30 11 1971
6 4 1971
4 8 2001
29 2 2001
32 11 2005
30 11 2004
6 9 1901

Sample output
Tuesday
Tuesday
Saturday
Incorrect Date
Incorrect Date
Tuesday
Friday

Problem information
Author: Jordi Petit
Translator: Carlos Molina
Generation: 2010-07-30 10:16:27

http://www.jutge.org