# How to determine the day of the week, given the month, day and year

First a brief explanation: In the Gregorian Calendar, over a period of four hundred years, there are 97 leap years and 303 normal years. Each normal year, the day of January 1 advances by one; for each leap year it advances by two.

$$303 + 97 + 97 = 497 = 7 * 71$$

As a result, January 1 year $N$ occurs on the same day of the week as January 1 year $N + 400$. Because the leap year pattern also recurs with a four hundred year cycle, a simple table of four hundred elements, and single modulus, suffices to determine the day of the week (in the Gregorian Calendar), and does it much faster than all the other algorithms proposed. Also, each element takes (in principle) only three bits; the entire table thus takes only 1200 bits; on many computers this will be less than the instructions to do all the complicated calculations proposed for the other algorithms.

Incidental note: Because 7 does not divide 400, January 1 occurs more frequently on some days than others! Trick your friends! In a cycle of 400 years, January 1 and March 1 occur on the following days with the following frequencies:

| | Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|---|
| Jan 1: | 58 | 56 | 58 | 57 | 57 | 58 | 56 |
| Mar 1: | 58 | 56 | 58 | 56 | 58 | 57 | 57 |

Of interest is that (contrary to most initial guesses) the occurrence is not maximally flat.

In the Mathematical Gazette, vol. 53,, pp.127-129, it is shown that the 13th of the month is more likely to be a Friday than any other day.The author is a 13 year old S.R.Baxter.

The Gregorian calendar was introduced in 1582 in parts of Europe; it was adopted in 1752 in Great Britain and its colonies, and on various dates in other countries. It replaced the Julian Calendar which has a four-year cycle of leap years; after four years January 1 has advanced by five days. Since 5 is relatively prime to 7, a table of 4 * 7 = 28 elements is necessary for the Julian Calendar.

There is still a 3 day over 10,000 years error which the Gregorian calendar does not take into account. At some time such a correction will have to be done but your software will probably not last that long!

Here is a standard method suitable for mental computation:

1. Take the last two digits of the year.
2. Divide by 4, discarding any fraction.
3. Add the day of the month.
4. Add the month's key value: JFM AMJ JAS OND 144 025 036 146
5. Subtract 1 for January or February of a leap year.
6. For a Gregorian date, add 0 for 1900's, 6 for 2000's, 4 for 1700's, 2 for 1800's; for other years, add or subtract multiples of 400.
7. For a Julian date, add 1 for 1700's, and 1 for every additional century you go back.
8. Add the last two digits of the year.
9. Divide by 7 and take the remainder.

https://blog.artofmemory.com/how-to-calculate-the-day-of-the-week-4203.html

Now 1 is Sunday, the first day of the week, 2 is Monday, and so on.

The following formula, which is for the Gregorian calendar only, may be more convenient for computer programming. Note that in some programming languages the remainder operation can yield a negative result if given a negative operand, so mod 7 may not translate to a simple remainder.

$$W = (k + \lfloor 2.6m - 0.2 \rfloor - 2C + Y + \lfloor Y/4 \rfloor + \lfloor C/4 \rfloor) mod 7$$

where $\lfloor \ \rfloor$ denotes the integer floor function,

$k$ is day (1 to 31)
$m$ is month (1 = March, ..., 10 = December, 11 = Jan, 12 = Feb) Treat Jan & Feb as months of the preceding year
$C$ is century (1987 has $C$ = 19)
$Y$ is year (1987 has $Y$ = 87 except $Y$ = 86 for Jan & Feb)
$W$ is week day (0 = Sunday, ..., 6 = Saturday)

Here the century and 400 year corrections are built into the formula. The $\lfloor 2.6m - 0.2 \rfloor$ term relates to the repetitive pattern that the 30-day months show when March is taken as the first month.

The following short C program works for a restricted range, it returns 0 for Monday, 1 for Tuesday, etc.

```
dow(m,d,y){y-=m<3;return(y+y/4-y/100+y/400+"-bed=pen+mad."[m]+d)%7;}
```

The program appeared was posted by sakamoto@sm.sony.co.jp (Tomohiko Sakamoto) on comp.lang.c on March 10th, 1993.