Remember the Linux `cal` command to display a calendar?

Let's make a program just like it.... something that may be appropriate for the coming new year.

```
$ cal 1 2009
    January 2009
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

Normally, people shouldn't be re-inventing the wheel, but we'll make an exception as we are still learning our way around Linux and C and many other stuff.

Your first task is to use a loop to print the days. Of course, any type of loop can be used, but a for-loop might be the easiest. Input the number of days (28, 29, 30 or 31), and print the dates in 7 columns, like this.

```
How many days? 28
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
```

Hint: The modulo or remainder operator (%) for integers might come in handy to determine when to print the newline char '\n'. Note that the numbers in the last column are all divisible by 7.

Wait... not all months start on a Sunday. Your second task is to make that a variable, say, the number of columns to skip before we print day 1.

```
How many days? 31
How many columns to skip? 4
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

Hint: You can still use the modulo operator; note that the numbers in the last column when added to the skip value are all divisible by 7.

Can't we just input the month and year and come up with an algorithm to determine the number of days? Good idea. Determining the number of days is fairly easy with simple rules – 31 days when the month number is 1, 3, 5, 7, 8, 10, 12; there are 28 days in February, except when it is a leap year where there are 29 days; and 30 days for the rest of the months. A multiple branch is perfect for this.

Hint: Use if-else if-else, or the switch-statement. Another hint: Years that are divisible by 4 are leap years, but there are rare exceptions. Most century years like 1900, 2100, 2200, 2300, ... are not leap years, except when the century year is divisible by 400. So the years 2000, 2400, 2800, ... are leap years.

```
Input month and year: 1 2009
How many columns to skip? 4
    January 2009
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

You might also want to put in some input-validation code like when the month is outside the range 1-12.

```
Input month and year: 13 2008
illegal month value: use 1–12
```

You can even calculate the number of days to skip with an ingenious algorithm:
Calculate the day of the week in 45 characters of code. Written by Mike Keith.

```
skip=(d+=m<3?y--:y-2,23*m/9+d+4+y/4-y/100+y/400)%7;
```

Reference: http://www.di-mgt.com.au/cprog.html

In case you are wondering about the extra operators in this expression, here is a translation in plain C:

```
   if (m < 3) {  // d = day, m = month, y = year
     d = d + y;
     y = y - 1;
   }
   else {
     d = d + y - 2;
   }
   skip = (23*m/9 + d + 4 + y/4 - y/100 + y/400) % 7;
```

```
Input month and year: 1 2009
    January 2009
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

Bonus: How do we use command line arguments?

```
#include <stdio.h>
#include <stdlib.h>
main( int argc, char *argv[] )
{
  int month, year;
  ...
  if (argc == 3) { // no. of command line arguments
    month = atoi( argv[1] );  // convert strings to ints
    year = atoi( argv[2] );
  }
  ...
}
```

Now we (almost) have the real thing...

```
$ ./ourcal 1 2009
    January 2009
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

jmsamaniego@uplb.edu.ph (2008)