Problem 4. Valid Date

Program: `isvalid.py`

Write a program named `isvalid.py` that accepts a date as input and write to the output whether the date is valid or not. Your program should accept three command-line arguments: `d` (day), `m` (month), and `y` (year). For m, use `1` for January, `2` for February, and so forth. Obvious examples of *invalid* dates 11 Nov 2019 and Jun 31 31 2019.

One of the things you should consider when checking a date's validity is whether the year is a leap year. In a leap year the month of February is 29 days instead of the usual 28. To check whether a year is a leap year, you can use the following sequence of Boolean formulas:

```
isLeapYear = (y % 4 == 0)
isLeapYear = isLeapYear and ((year % 100) != 0)
isLeapYear = isLeapYear or  ((year % 400) == 0)
```

Figure 1: A code fragment to test whether an integer corresponds to a leap year in the Gregorian calendar. A year is a leap year if it is divisible by 4 (2004), unless it is divisible by 100 in which case it is not (1900), unless it is divisible by 400 in which case it is (2000).

Using the above, one procedure to validate a data is as follows:

1. Check if the day (`d`) falls between 1-31.
2. Check if the month (`m`) falls between 1-12.
3. Check if the year (`y`) is a leap year.
4. If above checks are passed then you need to validate that the day falls within the specific month ranges. There are 3 types of months based on the day ranges:
    a. Months for which days are between 1-30.
    b. Months for which days are between 1-31.
    c. February for which days are between 0-28, unless it is a leap year then the range is between 0-29.

```
% python isvalid.py 4 11 2019
4/11/2019 is a valid date.


% python dayofweek.py 29 2 2019
29/2/2019 is an invalid date.

% python dayofweek.py 29 2 2020
29/2/2020 is a valid date.
```

Figure 2: Sample output for Problem 1

Problem 5. Checksums

Program: `checksum.py`

The International Standard Book Number (ISBN) is a 10-digit code that uniquely specifies a book. The rightmost digit is a *checksum* digit that can be uniquely determined from the other 9 digits, from the condition that $d_1 + 2d_2 + 3d_3 + \cdots + 10d_{10}$ must be a multiple of 11 (here d_i denotes the ith digit from the right). The check sum digit d_i can be any value from 0 to 10: the ISBN convention is to use the character ‘X’ to denote 10. Example: The checksum digit corresponding to 020131452 is 5, because 5 is the only value of m between 0 and 10 for which

$$10.0 + 9.2 + 8.0 + 7.1 + 6.3 + 5.1 + 4.4 + 3.5 + 2.2 + 1.m$$

is a multiple of 11. Write a program named `checksum.py` that takes a 9-digit integer as a command line argument, computes the checksum, and writes the ISBN number.

The output of the program should look like the following:

```
% python checkshum.py 020131452
The checksum value is 5
The ISBN-10 is 0-201-31452-5

% python checkshum.py 933257743
The checksum value is 9
The ISBN-10 is 9-332-57743-9
```

FIGURE 3: SAMPLE DIALOG FOR THE PROGRAM OF PROBLEM 5

Acknowledgements

Preparation of this problem set would not have been possible without adaptation from (Sedgewick, Wayne, & Dondero, 2015). The author gratefully acknowledges the work of the authors cited while assuming complete responsibility for any mistake introduced in the adaptation.

References

Sedgewick, R., Wayne, K., & Dondero, R. (2015). *Introduction to Programming in Python* (1st ed.). Addison-Wesley Professional.