Birthday POW

Problem Statement:

Given a birthday, how do you find the day of the week it was? Our process must be able to simply show a person how to find the date they were born on, or any other day they choose between 1901 and 2099. We must develop and define a solution, after being given a calendar for October 2023, the number of days per month, and some example birthdays and their days, and present it to others in a way that is simple to understand and is functional. Our solution can only use the 4 basic functions - addition, subtraction, multiplication, and division (no mods unless explicitly defined in the process).

Process:

Our final method:

Below is our final method. Early on in attempting to find this method, we had decided that our objective and approach would be finding how many days before a reference date (that you knew the day of the week for) that a given birthday was and then modding that number by 7 to find out how many days to count backwards from the day of the week of the reference day. We also determined quickly that finding how many days were between the reference date and a given birthday would be much more easily done in different steps/sections. Since we knew we had to factor in leap years into deriving this number of days, we first found how to find the number of days between the current date and the start of the year (with a condition for if the current year was a leap year) so that going forward, the next few steps would only have to worry about the year and not the specific date in the year, as they would be based off the starts of years. To better handle leap years, we then decided to have two extra points of data that would need to be given, these being the closest leap year before the birthday (or the birthday year if it was a leap year) and the closest leap year before the current date. This was done so because by having these, it would be easy to derive how many leap years had passed, as it would be one quarter of the amount of years between these two “anchoring” leap years. Using this new data, we were able to find how to derive the amount of days between the start of the leap year closest to the current date and the start of the current year, and then easily derive the amount of days between the two “anchoring” leap years. Since the second “anchoring” leap year was before the target birthday date and the process was attempting to find the amount of days before the current date that the birthday date was, we then had to backtrack to undo the overshoot.

Steps so far have achieved:

|<—x——s

| = leap year before target birthday date

x = target birthday date

s = current date (starting point)

Remaining steps need to achieve:

|—>x

(Undo overshoot to go back to target birthday date)

In order to undo the overshoot, we did much of the same process for finding from the current date to the start of the current year and then the nearest leap year. We first found how many days forward from the nearest leap year before the target birthday date that the actual target birthday year was, and then how

many more days forward to get from the start of the target birthday year to the actual birthday date. Keeping track of all of the days backward and then forward, we then defined how to “mod” the number by 7 to get rid of the whole weeks that passed (as moving whole number weeks at a time would not change the day of the week) and find how many days backward from the current day of the week there were. Then, we just had to count that many days back from the current day of the week to find the day of the week of the given birthday.
After defining this process, we coded it for easier testing and for a more well defined formula as well as tested it manually during the XYZ groups day to ensure the clarity of the process and verify its accuracy.

Info needed:
Current date - year, month, day, day of the week (0-6 where Sunday is 0)
Target date/birthday - year, month, day
Nearest leap year before the birthday date
Nearest leap year before the current date
If the current and birthday years are leap years

Definitions:
A year is a leap year if it is a multiple of 4.

Modulo operator (%) is defined as the following:
x % y = x - y * (x / y rounded toward 0, meaning round up if negative and round down if positive)

1. Finds days from current date to beginning of the current year (Jan 1):
    - Take what number day the current day is in the month, and add it to the number in the following table corresponding to the current month:
      [Jan: 0, Feb: 31, Mar: 59, Apr: 90, May: 120, Jun: 151, Jul: 181, Aug: 212, Sep: 243, Oct: 273, Nov: 304, Dec: 334]

2. Factor in if the current year is a leap year:
    - If it is currently a leap year, and the current month is later than February, add 1 to the number you have.
    - If the current year is **not** a leap year, find the difference between the current year and the nearest leap year before the current year, multiply this by 365, add 1, and add it to the number you have.
        - (currentYear - nearestLeapYearBeforeCurrent) * 365 + 1

3. Find how many days it is from the start of the nearest leap year before current year to the start of the nearest leap year before the birthday year:
    - Multiply 365 * 4 + 1 by (the difference between the closest leap year before the current year and the closest leap year before the birthday date, divided by 4), and add this to your number.
        - (365 * 4 + 1) * (nearestLeapYearBeforeCurrent - nearestLeapYearBeforeTarget) / 4

4. Find how many days it is from the start of the nearest leap year before the birthday year to the beginning of the birthday year, and subtracts it previous number (since the previous number represents

going from after the birthday year to the leap year before the birthday year, you need to backtrack to the birthday year, thus subtracting):

- If the birthday date's year is not a leap year, subtract (365 multiplied by the difference between the birthday year and the closest leap year before it + 1) from your number.
- (365 * (birthdayYear - nearestLeapYearBeforeTarget) + 1)

5 & 6. These 2 steps find how many days it is from the start of the birthday year to the birthday date, and subtracts it from the previous number for the same reason as the previous step:

5. Take what number day the birthday day is in the month +1, and subtract it from your number.

6. Using the following table again, subtract the number that corresponds to the birthday month from your number.

[Jan: 0, Feb: 31, Mar: 59, Apr: 90, May: 120, Jun: 151, Jul: 181, Aug: 212, Sep: 243, Oct: 273, Nov: 304, Dec: 334]

7. Account for if the birthday year is a leap year (again):

- **If** the target year is a leap year and the target month is after February, subtract 1 more from your number.

8. This step gets rid of the "whole weeks" and leaves only the remainder of days (since going back any number of whole weeks keeps you at the same day of the week)

- Divide the number by 7 and find the remainder (number % 7)

9. Get the actual day of the week:

- Count that many days before (if the evaluated value is negative) or after (if positive) the current day of the week to obtain the day of the week of the birthday date.

One other method we tested was finding the starting day of each year then calculating the day of the week based on that day. After finding the starting day of the year it is fairly simple, but overall it is more complicated to find the starting day of the year than the method we have described above and have our solution to below. When working on this method we also looked at patterns of the first day of each year but eventually this pattern did not help in finding a solution. We also found that if you know the starting day of the month, if you do (day-1)mod7, it corresponds to the number of days to add. However, we did not end up using this in our solution either (it uses mod and also wasn't helpful later).

Solution

Definitions:
A year is a leap year if it is a multiple of 4

Modulo operator (%) is defined as the following:
x % y = x - y * (x / y rounded toward 0, meaning round up if negative and round down if positive)

1. Take what number day the current day is in the month, and add it to the number in the following table corresponding to the current month:
   [Jan: 0, Feb: 31, Mar: 59, Apr: 90, May: 120, Jun: 151, Jul: 181, Aug: 212, Sep: 243, Oct: 273, Nov: 304, Dec: 334]
2. If it is currently a leap year, and the current month is later than February, add 1 to the number from step 1. To find if it is currently a leap year, divide the year by 4. If there is no remainder, it is a leap year, otherwise it is not a leap year. Otherwise, if the current year is not a leap year, find the difference between the current year and the nearest leap year before the current year, multiply this by 365, add 1, and add it to the number from step 1. To find the closest leap year before the current year, divide the current year by 4 and find the remainder. Then subtract this remainder from the current year to find the closest leap year before the current year.
   (currentYear - nearestLeapYearBeforeCurrent) * 365 + 1
3. Multiply 365 * 4 + 1 by (the difference between the closest leap year before the current year and the closest leap year before the birthday date, divided by 4), and add this to your number from step 2.
   (365 * 4 + 1) * (nearestLeapYearBeforeCurrent - nearestLeapYearBeforeTarget) / 4
4. If the birthday date's year is not a leap year, subtract (365 multiplied by the difference between the birthday year and the closest leap year before it + 1) from your number from step 3.
   (365 * (birthdayYear - nearestLeapYearBeforeTarget) + 1)
5. Take what number day the birthday day is in the month, add 1, and subtract this number from your number from step 4.
6. Using the following table again, subtract the number that corresponds to the birthday month from your number from step 5.
   [Jan: 0, Feb: 31, Mar: 59, Apr: 90, May: 120, Jun: 151, Jul: 181, Aug: 212, Sep: 243, Oct: 273, Nov: 304, Dec: 334]
7. If the birthday year is a leap year and the birthday month is after February, subtract 1 more from your number from step 6.
8. Divide the number from step 7 by 7 and find the remainder (find your number % 7).
9. Count that many days given by step 8 before (if the evaluated value is negative) or after (if positive) the current day of the week to obtain the day of the week of the birthday date.

We know our solution is correct and complete because after having tested it ourselves and with peer reviewers it works for all dates that we tested.

Extensions

CODE: Descriptions and whatnot highlighted in green in the middle:
Here is an online Java compiler in case:
https://www.w3schools.com/java/tryjava.asp?filename=demo_compiler

```java
import java.util.HashMap;
import java.util.Scanner;

public class BirthdayPOW {

public static void main(String[] args) {

Scanner birthdayWizard = new Scanner(System.in);

int DAYYY;
int daysFromNow;

System.out.println("Please enter all inputs as integers :)");
System.out.print("What is the current year: ");
int currYear = birthdayWizard.nextInt();

System.out.print("\nWhat is the current month: ");
int currMonth = birthdayWizard.nextInt();

System.out.print("\nWhat is the current day in the month: ");
int currDayInMonth = birthdayWizard.nextInt();

System.out.print("\nSunday is defined as 0, the start of the week\nWhat is the current
day in the week: ");
int currDay = birthdayWizard.nextInt();

boolean currIsLeapYear = currYear % 4 == 0;
int closestLeapYearToCurr = currYear - currYear % 4;

System.out.print("\nWhat is the birthday year: ");
int targetYear = birthdayWizard.nextInt();

System.out.print("\nWhat is the birthday month: ");
int targetMonth = birthdayWizard.nextInt();

System.out.print("\nWhat is the target day in the birthday month: ");
int targetDayInMonth = birthdayWizard.nextInt();

boolean targetIsLeapYear = targetYear % 4 == 0;
int closestLeapYearToTarget = targetYear - targetYear % 4;

int[] monthLens = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
```

```
int[] daysFromYearStart = new int[12];
daysFromYearStart[0] = 0;
for (int i = 1; i < monthLens.length; i++) {
daysFromYearStart[i] = daysFromYearStart[i - 1] + monthLens[i - 1];
}

daysFromNow = daysFromYearStart[currMonth - 1];
daysFromNow += currDayInMonth;

if (currIsLeapYear && currMonth > 2) {
daysFromNow++;
}
if (closestLeapYearToCurr != currYear) {
daysFromNow += 365 * (currYear - closestLeapYearToCurr) + 1;
}
daysFromNow += (365 * 4 + 1) * (closestLeapYearToCurr - closestLeapYearToTarget) / 4;
if (closestLeapYearToTarget != targetYear) {
daysFromNow -= 365 * (targetYear - closestLeapYearToTarget) + 1;
}
daysFromNow -= daysFromYearStart[targetMonth - 1];
daysFromNow -= targetDayInMonth;
if (targetIsLeapYear && targetMonth > 2) {
daysFromNow--;
}
daysFromNow %= 7;

DAYYY = currDay - daysFromNow;
if (DAYYY < 0) {
DAYYY += 7;
}
System.out.println(daysFromNow);
HashMap<Integer, String> daysOfWeek = new HashMap<>(7);
daysOfWeek.put(0, "Sunday");
daysOfWeek.put(1, "Monday");
daysOfWeek.put(2, "Tuesday");
daysOfWeek.put(3, "Wednesday");
daysOfWeek.put(4, "Thursday");
daysOfWeek.put(5, "Friday");
daysOfWeek.put(6, "Saturday");

System.out.println("\nYou were born on a " + daysOfWeek.get(DAYYY));
```

```
birthdayWizard.close();
}
}
```

In the future, this project could be extended to include all years beyond 1900 and 2100. This becomes more difficult because multiples of 100 are not leap years even though they are multiples of 4. We could also extend a solution to include multiple types of calendars, but this also becomes more difficult as each calendar has special rules that would most likely require separate solutions for each calendar type.