Nicholas Giza, Lindsey Paradise, Kayla Vallecillo, Cecilia Carbonell
10/13/23
Queens - Section B
Birthday Pow Write Up

Problem Statement

What do Mahatma Gandhi, Madonna, and Eleanor Roosevelt have in common? They were all born on Saturdays! As you start to think about what day of the week you were born on, it may be difficult to find out, especially without a reference such as a calendar or an online calculator. With factors such as leap years, shifting weekdays every year, and limited access to a calendar or reference, this process becomes strenuous and time-consuming. Through the use of arithmetic and logic, however, this process becomes easier and more organized. To make this process easy and accessible to *all* birthdays after 1901 and before 2100, a universal process that evaluates what day of the week that someone was born on must be created. This paper will outline and justify one mathematical approach to this problem.

Process

To alleviate any confusion, the team defined “day” as the day of the week (ex. Sunday) whereas “date” was the numerical day of the month (ex. 10). Early in our process, the team noticed that their 17th birthdays fell on the same day of the week as the days on which they were born. This led the team to make the false assumption that this pattern is true for all dates, 17 years apart. This was later proven incorrect, and the team had to quickly redirect to a new approach.

Following discussion, it was concluded that the solution should be thought of as two parts: the first part would find the first day of the year, while the second would find the change in days up to the target date. It was believed that once developed, those steps could be combined and return a value representing the day of the week that a target date was on. The team began by addressing the second phase. In order to pursue a solution mathematically, a system of date representations was devised where the days Sunday through Saturday were represented by the numbers 0 through 6, respectively. From there, a reference date was needed and through

counting backwards from the current date, the first day of the year 2023 was found to be a Sunday. It took a large amount of logical consideration and trial and error to figure out that the first day of the year should be represented by the number 1, and that every seven day cycle will end on the day before the starting day. For example, a seven day cycle that starts on a Sunday will end on a Saturday and January 6th is the 6th day of the year. Based on these discoveries, it was reasoned that the number of whole weeks, minus one (to compensate for a week cycle ending one day before the starting day), plus the number of surplus days, would equate to the numerical difference in days between the first day of the year and the current day/target day. This can also be written as the date number in mod 7, minus one [1], see appendix. The outputs and logic in both expressions are the same.

As the team tested the previous method, finding no further issues, the first step now had to be addressed. The subsequent process was the most challenging and yielded the most problems. The development of the first step began with some additional logical reasoning. The team noticed that the days repeated every 28 years, not 17, as was previously thought. It became apparent that if any year could be simplified to an equivalent year between 1901 and 1928, then the first day of the corresponding year could be solved for and would represent the first day of the target year. Unfortunately, this was incredibly difficult to structure into the form of an equation. Eventually, an error-prone equation [2] was created based on a few reference dates to represent this pattern. It used logic that claimed that the difference between the target year and 1901 could be divided by 28 and the remainder would represent the position in the 1901 to 1928 sequence. Then, since there was a 5 day increase in the starting day every four years (+1 day for each of the three normal years and +2 for the leap year), the position in the aforementioned sequence was multiplied by 5/4 and truncated to represent the total change in the starting day between 1901 and the target year. Here, +1 day meant that a year began one day later than the previous year. The number two was added to this value to account for the fact that January 1, 1901 was a Tuesday. All initial testing showed showed that this method worked. In order to further test this, the equation was inserted into a spreadsheet to test all years between 2023 and 1901. To the team's dismay, this logic failed to account for the dates in a leap year prior to February 29th, since the current model accounted for the extra +1 shift on January 1st, not February 29th. The whole idea had to be scrapped and the team began to search for a new direction.

After communicating with Gustavo, a member of a different group, the idea of having 365.25 days in a year was found to be crucial to the solution. There are, in fact, 365.25 days in a year, which is why there are leap years, so this approach would already account for them. With this in mind, the number of years since 1901 could be multiplied by the average days in a year to find an approximate number of days between the first day of 1901 and the first day of a target year. Truncation (removing any decimals) would simplify the numbers and the resulting value could be increased by 2 (because 1901 started on a Tuesday) and put in mod 7 (the remainder when divided by 7) to find the first day of the target year [3]. This approach was successful for all date cases (non-leap year, dates before February 29th in a leap year, and after February of a leap year).

Our two steps were then added together in mod 7, yielding our final solution [4] in the form of a numerical day value (0-6). Thorough testing of all types of dates proved our method to be correct. The group tested leap years, years prior to leap years, years after and everything in between to ensure one-hundred percent accuracy. Our steps were then broken down into friendlier language to universalize usability across all mathematical skill levels. After several failed attempts and many revisions, our solution was complete.

Solution

To find the day of the week of *m*/*d*/*y*:

1. Subtract 1901 from the target year (*y*).
2. Multiply the answer from step 1 by 365.25
3. Add 2 to the answer from step 2
4. Round down to the nearest whole number
5. Add the number next to the target month in Table One to *d*
6. Subtract 1 from the answer to step 5
7. Add the answer from step 4 to the answer from step 6
8. Divide the result by seven and record the remainder
9. Use Table Two to find the day of the week that corresponds to the remainder, that day is the day of the target date.

Table One

* A year is a leap year if it is evenly divisible by 4

Target Month	Days to Add to *d*	* Days to Add to *d* if *y* is a Leap Year
January	0	0
February	31	31
March	59	60
April	90	91
May	120	121
June	151	152
July	181	182
August	212	213
September	243	244
October	273	274
November	304	305
December	334	335

Table Two

Day of the Week	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Remainder	0	1	2	3	4	5	6

Extensions

Through the process of building a model to determine what day any date from 1901 to 2100 occurred, tools including arithmetic, logic, pattern recognition, and creativity were utilized. This toolset can be applied to a variety of additional problems also relating to other types of calendars and separate multi-annual cycles. Intriguing extensions to this project include:

1. Expanding the present model to account for years before 1900 and after 2100. This would allow users to see what day their birthday will be in the future and plan accordingly. If the model were to account for years before 1582, it would have to factor in the transition from the Julian to the Gregorian calendar.
2. Calculating future Congressional and Presidential election years. Bonus points if the model is able to determine when these elections years overlap.
3. Determining, for a given year, if the Summer Olympics, Winter Olympics, or both were hosted.
4. Tracking lunar cycles and calculating what phase of the moon it was or if it will be on a specified day.
5. Modeling the future dates of holidays that do not fall on the same number day each year. For example, since each year Easter Sunday is the first Sunday after the Pascal Full Moon, could extension 4 be further adapted to determine the date of Easter in future years? Additional holidays to model include Thanksgiving, Labor Day, Chinese New Year, and Indigenous Peoples' Day.

These extensions, ranging in difficulty, would create a new environment to build upon and refine the skills gained while solving the original birthday Problem of The Week.

Appendix

[1] (#th Day of Year)-1) mod 7

[2] ((truncate to units place(1.2*((Y-1901) mod 28))mod7)+2

[3] (truncate to units place((YEAR-1901)*365.25+2)) mod 7

[4] ((truncate to units place((YEAR-1901)*365.25+2)) + (#th Day of Year)-1)) mod 7

[5] Link to the working model in a Google Sheets file.