The Calendar Problem

Richard Hoshino

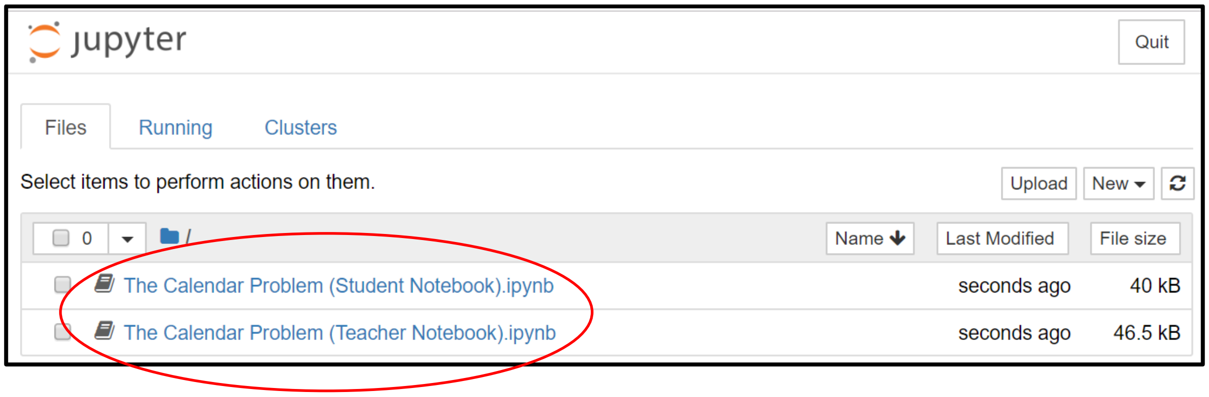
The Calendar Problem is a free resource for teachers and students, and is part of the Callysto Project (www.callysto.ca), a federally-funded initiative to bring computational thinking and mathematical problem-solving into Grade 5-12 Canadian classrooms.

During the 2018-19 school year, I had the fortune of visiting over a dozen schools and working with 700+ students, sharing rich math problems that incorporated the Callysto technology (a web-based platform known as a Jupyter Notebook, freely accessible to anyone with an Internet connection).

In this Callysto Notebook, we present The Calendar Problem, a lesson that was first taught to Grade 10 students at Howe Sound Secondary School in Squamish, and Grade 11 students at York House School in Vancouver.

To access this free Notebook, visit www.bit.ly/CallystoCalendar

This link will take you to a screen that looks like this.



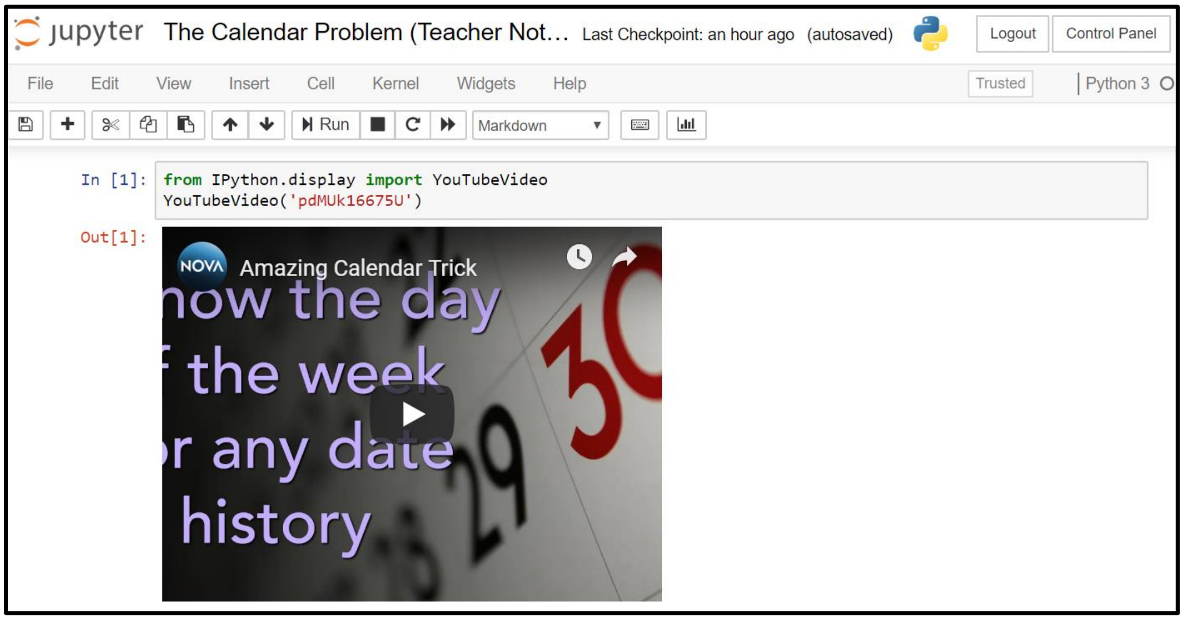
Select items to perform actions on them.

Name	Last Modified	File size
The Calendar Problem (Student Notebook).ipynb	seconds ago	40 kB
The Calendar Problem (Teacher Notebook).ipynb	seconds ago	46.5 kB

Students are to click on the Student Notebook, while this document lists the contents of the Teacher Notebook.

We begin by showing the following YouTube video, which presents a math professor performing an amazing "Calendar Calculation".

https://www.youtube.com/watch?v=pdMUk16675U



```
In [1]: from IPython.display import YouTubeVideo
        YouTubeVideo('pdMUk16675U')
Out[1]:
```

Make sure you cut off the video at the 1:00 mark, just before the professor explains HOW he performs his calculation. We will intentionally not show the professor's solution at any time during this lesson, since the students will reproduce their own algorithm themselves.

If your school blocks YouTube, then skip the video.

For your information, the video profiles a mathematician (or "mathemagician") named Arthur Benjamin, who asks an audience member for her birthday, and instantly calculates the day of the week of that particular date. For example,

Student: "I was born on May 25, 2004".

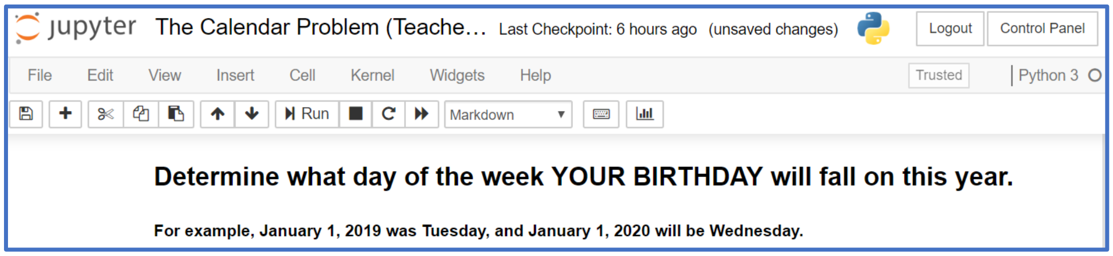
Arthur Benjamin: "That was a Tuesday".

Start this activity by having everyone line up along the wall, arranging themselves in order of their birthday, from January 1 on one end to December 31 on the other end. (Ignore the year of birth).

Form groups of four, where each group's four birthdays are as spread out as possible. For example, if there are 28 students in the class, have the students number themselves down the line, from One to Seven, and repeat until all students have a number. Thus, everyone with the same number will be on the same team for this activity, and ideally these four students will have their birthdays spaced (roughly) three months apart.

If the number of students is not a multiple of four, it's okay if some groups have three or five students.

Now ask the students the following question, which we'll call the Calendar Problem:



Determine what day of the week YOUR BIRTHDAY will fall on this year.

For example, January 1, 2019 was Tuesday, and January 1, 2020 will be Wednesday.

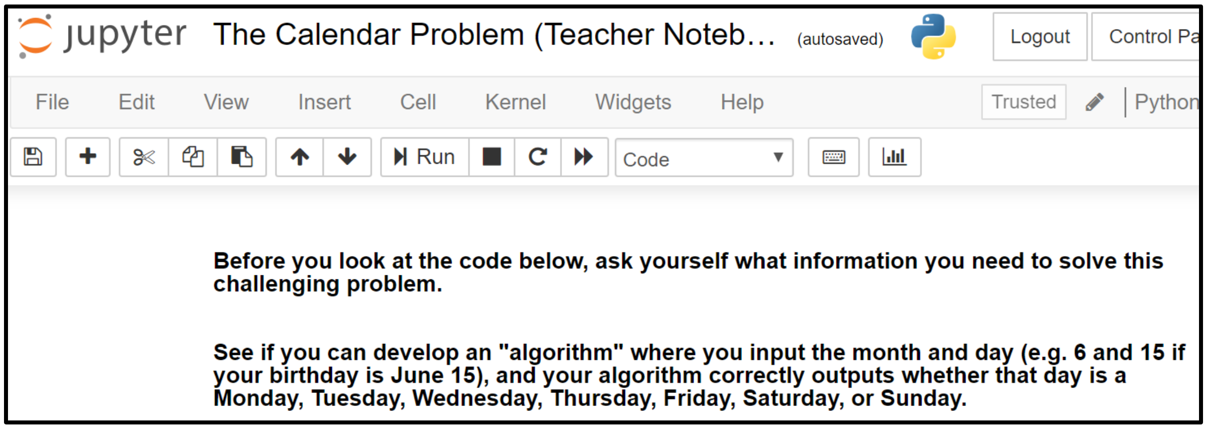
The students are to determine the day of the week that their birthday falls on THIS YEAR. They are to complete this activity in their assigned 4-person groups, and work together to determine the correct day of week for everyone in the group.

Tell them the day of the week for January 1 (New Year's Day), so they have a reference point from which they can perform their calculations. Here are the correct days for January 1 for the next few years:

- 2019=Tuesday
- 2020=Wednesday
- 2021=Friday
- 2022=Saturday

For the purposes of this Teacher Manual, I will assume that we are working in the Year 2019. If you are teaching this lesson in the Year 2020, please make the appropriate adjustments, changing each instance of "Tuesday" to "Wednesday" by adding one day.

Here are two prompts to get the students going:



Before you look at the code below, ask yourself what information you need to solve this challenging problem.

See if you can develop an "algorithm" where you input the month and day (e.g. 6 and 15 if your birthday is June 15), and your algorithm correctly outputs whether that day is a Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, or Sunday.

One student in the class will surely know (or remember) the number of days in each month:

- January = 31 days
- February = 28 days normally, and 29 in a leap year
- March = 31 days
- April = 30 days
- May = 31 days
- June = 30 days
- July = 31 days
- August = 31 days
- September = 30 days
- October = 31 days
- November = 30 days
- December = 31 days

Several other students will note that January 1 must be the same day of week as January 8, January 15, January 22, and January 29. Furthermore, it's no surprise that each of these numbers (1, 8, 15, 22, 29) give the same remainder when divided by 7.

Thus, some birthdays will be relatively easy to calculate. For example, if a student is born on January 17, then they use the fact that January 1 is a Tuesday to infer that January 15 is a Tuesday, and then determine that January 16 is a Wednesday, and conclude that January 17 must be a Thursday.

Some students might notice that for birthdays in the month of January, the answer can be found by simply taking the date, dividing by 7, and calculating the remainder. Then they can use this table to read off the answer:

- 0 = Monday
- 1 = Tuesday
- 2 = Wednesday
- 3 = Thursday
- 4 = Friday
- 5 = Saturday
- 6 = Sunday

Here are two common approaches for solving the Calendar Problem.

Approach One:

Count the number of days that have elapsed from the start of the year (January 0) until the target date. For example, March 23 consists of 31+28+23 days, since we need to add up all the days in January and February and then the twenty-three days in March. This adds up to 82. We divide by 7. Since 82 = 7×11 + 5, the remainder is 5. From the above table, we see that a remainder of 5 corresponds to Saturday.

Approach Two:

Determine the day of the week for the first date of each month, showing that if January 1 falls on Tuesday, then February 1 must be a Friday, March 1 must also be a Friday, and so on. From this, students can solve their problem for any given date by adding or subtracting increments of seven. For example, March 23 has to be the same date as March 16, March 9, and March 2 – and so March 23 has to be a Saturday, since March 1 is a Friday.

A clever approach combines these two paradigms, using the first date of each month to determine the appropriate "shift". For example, March 1 is 31+28 = 59 = 8×7+3 days after January 1, and so March 1 is "shifted" by 3 days compared to January 1. Thus, if we know that the shift number of March is 3, then we can determine the day of week of March 23 by adding the date to the shift number (3+23=26=3×7+5), dividing the number by 7 and taking the remainder (which is 5), and then reading the above table to conclude that the answer is Friday. Unless someone has determined this solution themselves, I recommend the teacher present this "shifting algorithm" to the class, walking the students through the key steps.

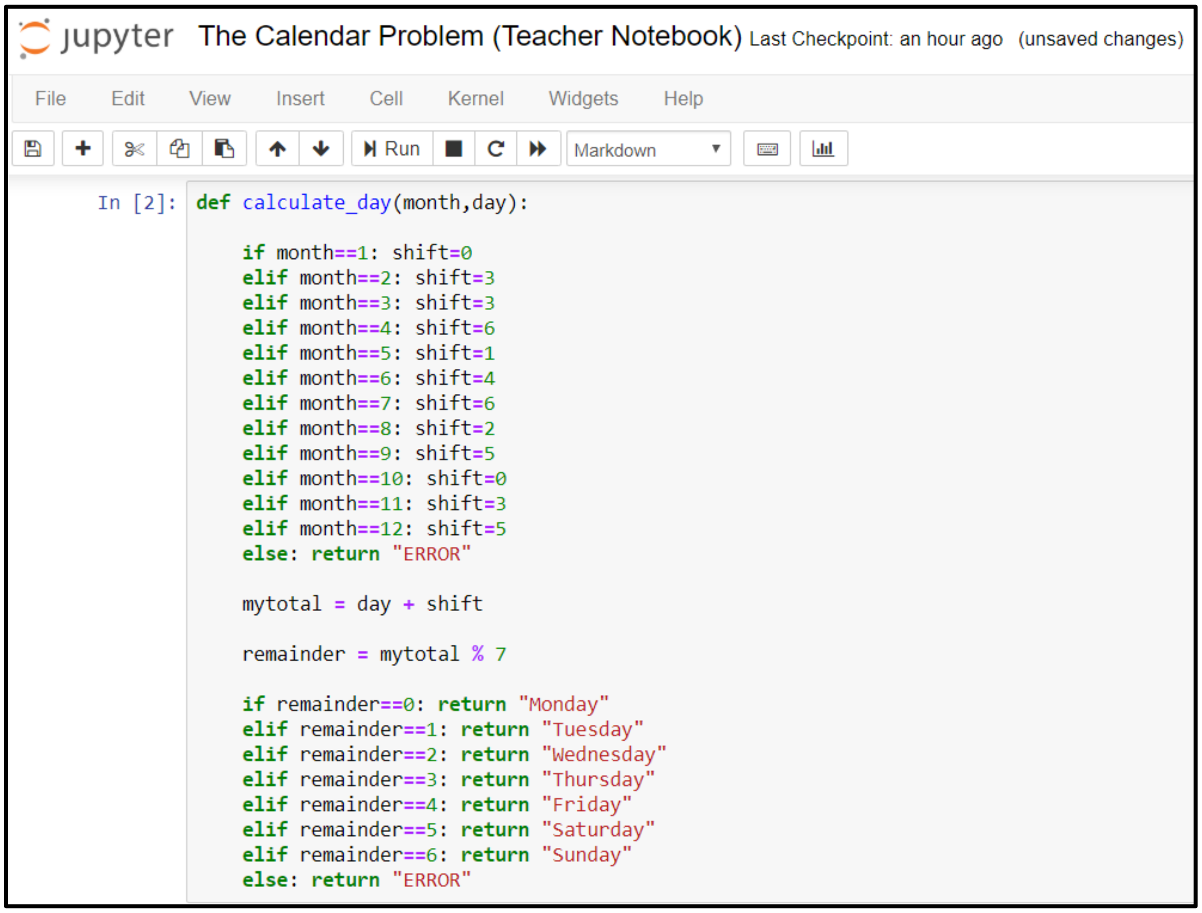
The students can determine the shift dates of each month, and they should get the following table:

Jan = 0, Feb = 3, Mar = 3, Apr = 6, May = 1, Jun = 4
Jul = 6, Aug = 2, Sep = 5, Oct = 0, Nov = 3, Dec = 5

For example, the shift number for June is 4, since the number of days until the start of June is 31+28+31+30+31=151=21×7+4, which has a remainder of 4 upon division by 7.

In other words, June 1 is exactly 21 weeks and 4 days after January 1 which implies that the shift for June is 4.

Now have the students read this code block, which is written in the Python programming language. They can either read this on their own computer or on the classroom projector. Notice how the shift numbers are identical to what is written above.


jupyter
The Calendar Problem (Teacher Notebook) Last Checkpoint: an hour ago (unsaved changes)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Markdown
In [2]:

```python
def calculate_day(month,day):

    if month==1: shift=0
    elif month==2: shift=3
    elif month==3: shift=3
    elif month==4: shift=6
    elif month==5: shift=1
    elif month==6: shift=4
    elif month==7: shift=6
    elif month==8: shift=2
    elif month==9: shift=5
    elif month==10: shift=0
    elif month==11: shift=3
    elif month==12: shift=5
    else: return "ERROR"

    mytotal = day + shift

    remainder = mytotal % 7

    if remainder==0: return "Monday"
    elif remainder==1: return "Tuesday"
    elif remainder==2: return "Wednesday"
    elif remainder==3: return "Thursday"
    elif remainder==4: return "Friday"
    elif remainder==5: return "Saturday"
    elif remainder==6: return "Sunday"
    else: return "ERROR"
```

At this point, ask the students to explain to each other what is happening, to see how much of this code they can understand, even though for many students this Python program will be written in a foreign language.

After the students have gone through this process, we then walk through each section of this code as a class, ignoring the syntax (e.g. if-elif-else, writing == instead of =) and just focusing on what is happening at each step.

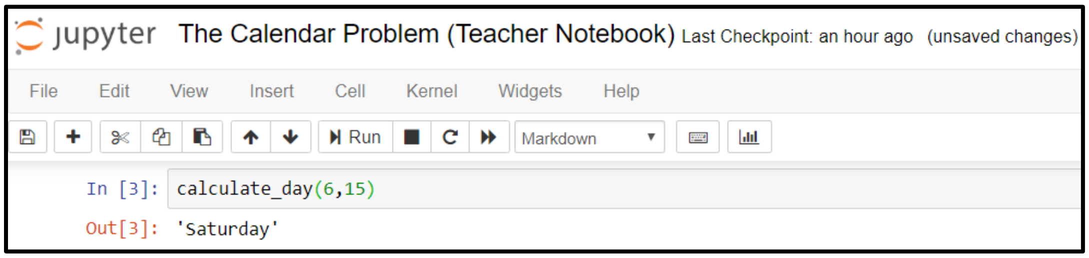
- We first define a function called **calculate_day** that accepts two inputs: *month* and *day*. For example, if your birthday is June 15, then the inputs are *month*=6 and *day*=15.
- We then use the *month* variable determine the shift number, which we call *shift*.
- We then define a variable called *mytotal* that adds *day* and *shift*.
- We then define a variable called *remainder* that divides *mytotal* by 7, and calculates the remainder. (That's what the %7 symbol does).
- We then use the remainder to output the correct day of week.

Although this is being presented in a Mathematics class (rather than a Computer Programming class), the students should be able to follow what is being done in this Python program, since they created this algorithm themselves.

In fact, through this process of solving the Calendar Problem and determining an algorithm that works for any birthday, the students demonstrate the four principles of the **Computational Thinking process**.

- Decomposition: break down the problem into smaller tasks
- Pattern recognition: identify similarities, differences, and patterns within the problem
- Abstraction: identify general principles and filter out unnecessary information
- Algorithm design: identify and organize the steps needed to solve the problem

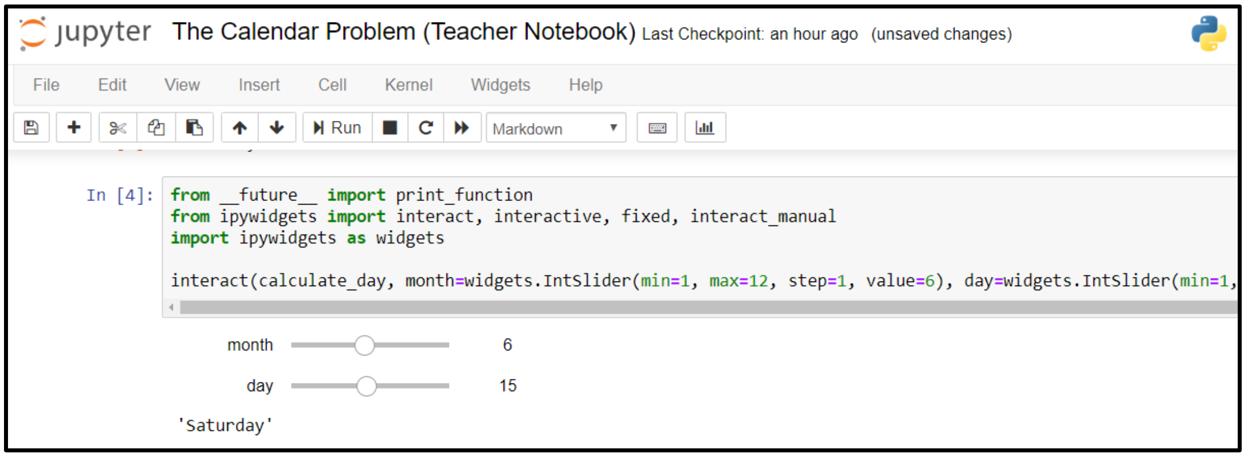
Now have students verify that this program works, by inputting their birth month and birth date, and comparing their calculated Day of Week to the answer below.


jupyter The Calendar Problem (Teacher Notebook) Last Checkpoint: an hour ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run Markdown

```
In [3]: calculate_day(6,15)
Out[3]: 'Saturday'
```

This Callysto Notebook has a fancy “widget” tool that enables students to determine the result of the **calculate_day** function without inputting their birth month and date. All they need to do is slide the two widgets to the correct month and date, and the answer will be calculated instantly.

In a Math class, we recommend ignoring the Python code that enables this widget tool to work. On the other hand, it might be worthwhile to explore this code block with a Programming class, going through the code line by line.


jupyter The Calendar Problem (Teacher Notebook) Last Checkpoint: an hour ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run Markdown
month
6
day
15

```
In [4]: from __future__ import print_function
        from ipywidgets import interact, interactive, fixed, interact_manual
        import ipywidgets as widgets

        interact(calculate_day, month=widgets.IntSlider(min=1, max=12, step=1, value=6), day=widgets.IntSlider(min=1,

'Saturday'
```

For a strong class (say a group of Grade 11s or 12s), you can generalize this problem to any birth year, so that the students can determine the day of week for their own birthday (e.g. June 15, 1978 was a Thursday). To do this, divide the participants via the “jigsaw method”, where each four-person team is assigned a letter (A, B, C, D), and the new letters form the new groups. For example, if there are 24 students in the class, the six groups of four become four groups of six.

And in these new groups, students solve the same problem, replacing the year 2019 with their birth year. In solving this harder problem, students will realize that each 365-day year contributes one extra day (52 weeks plus 1 day). Thus, if January 1, 2019 is a Tuesday, then January 1, 2018 is a Monday. In other words, when we go from 2019 to 2018, we simply subtract 1, to go from Tuesday to Monday.

Students can use the information they know (their day of week for 2019) and work backwards until they find the correct answer for their year of birth, taking leap years into account.

During one of our school visits (in 2018), one student made the powerful insight that her birthday in 2001 must be the same day of week as her birthday in 2018, since there are 17 “extra days” in addition to the four Feb 29 “leap days” that occurred in 2004, 2008, 2012, and 2016. Since 17+4=21, the calendar shifted 21 days between her birthday in 2001 and her birthday in 2018. And since 21 is a multiple of 7, if her birthday fell on a Tuesday in 2018, then it must have fallen on a Tuesday in 2001. This is how to handle the tricky concept of leap years.

And a different student observed that the calendar repeats itself every 28 years, since each year contributes one extra day (52 weeks plus 1 day), and there are 7 occurrences of February 29 during any 28-year period. Thus, the calendar shifts by 28+7 = 35 days, which is a multiple of 7.

For super-keen students, encourage them to extend this algorithm to other famous dates in world history, dating back centuries.

Some examples are

- June 6, 1944 (D-Day)
- July 1, 1867 (Confederation Day in Canada)
- July 4, 1776 (Independence Day in the USA)
- April 23, 1616 (Death of William Shakespeare)
- April 23, 1564 (Birthday of William Shakespeare)

Students will have to be careful about ensuring the correct calculation of leap years, due to the quirky rules that occur when the year is a multiple of 100 but not a multiple of 400. Specifically, the years 1600 and 2000 are leap years, while the years 1700, 1800, 1900 are not leap years.

For your strongest students, here is one final challenge problem:

Create your own algorithm to determine the correct day of week for any date in the 20th or 21st century (Jan 1, 1901 to Dec 31, 2100). Your algorithm should be one that you can compute in less than twenty seconds, either in your head or by jotting down some calculations on a sheet of paper. Present your algorithm, and clearly and carefully justify WHY your algorithm outputs the correct day of week, no matter what input date is chosen. Finally, verify your algorithm using the date October 29, 1929, a famous date in world history known as "Black Tuesday".

I will purposely not post the solution to that challenge problem here. However, your students are encouraged to write out their solution, scan it as a PDF, and then e-mail it to me (richard.hoshino@gmail.com). I would be happy to write to the student and provide feedback on their solution.

About the Author

Richard Hoshino is a mathematics professor at Quest University Canada, an innovative liberal arts and sciences university located in Squamish, BC. He is the author of "The Math Olympian", has published 30+ research papers across numerous fields, and runs a thriving math consulting business that inspires students, engages teachers, and optimizes businesses. Richard was the 2017 recipient of the Adrien Pouliot Award, awarded by the Canadian Mathematical Society as a lifetime achievement award to celebrate "significant and sustained contributions to mathematics education in Canada". For more information on Richard and his work, please visit www.richardhoshino.com.