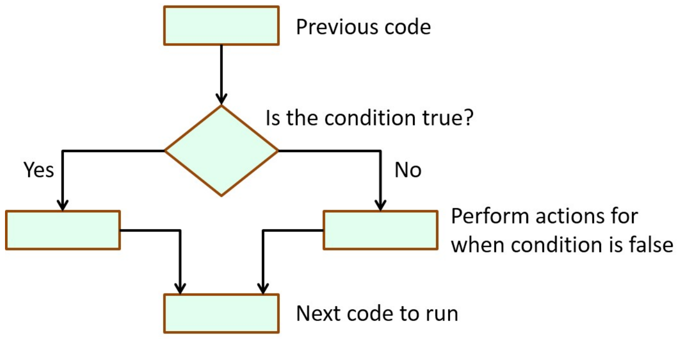
420-201-RE Assignment 2-B Due on 3 October 2016

Objectives

1. Practice using relational and logical operators
2. Practice using **switch** construct
3. Practice using appropriate conditional construct for a given task



Your Task: Convert Day of Year $\Longrightarrow$ Date

Day of year is a number ranging from 1 through 365 (366 on leap years). January 1 is day 1. Write a program that prompt for and reads a year followed by a day number in that year. Your program should convert the given year and day numbers to a date.

Sample runs of your program should produce the following output:

```
Enter a year and a day number in that year: 2000 200
Tuesday, July 18, 2000 is day 200
```

```
Enter a year and a day number in that year: 2012 365
Sunday, December 30, 2012 is day 365
```

```
Enter a year and a day number in that year: 2013 365
Tuesday, December 31, 2013 is day 365
```

```
Enter a year and a day number in that year: 2014 366
Error: day number cannot exceed 365 for the year 2014
```

```
Enter a year and a day number in that year: y2k 200
Error: invalid year y2k
Error: year value must be a positive integer beyond 1582
```

```
Enter a year and a day number in that year: 1000 365
Error: invalid year 1000
Error: year value must be a positive integer beyond 1582
```

```
Enter a year and a day number in that year: 2000 ten
Error: invalid day number ten
Error: day number must be a positive integer from 1 through 366 in the year 2000
```

```
Enter a year and a day number in that year: 2001 -21
Error: invalid day number -21
Error: day number must be a positive integer from 1 through 365 in the year 2001
```

Background

Gregorian calendar

A **Gregorian calendar** date runs from 1 January to 31 December of a year beyond 1582. Common years have 365 days but leap years have 366 days; leap years add a 29th day to February, which normally has 28 days.

Valid Gregorian Dates

A valid Gregorian date (*day*, *month*, *year*) satisfies *all* of the following conditions:

(a) the *year* value is greater than 1582.
(b) the *month* value is in the range 1 through 12, and
(c) the *day* value is in the range 1 through *either* 28, 29, 30, or 31, depending on the given *month* and *year*.

Days in Months

A Gregorian year is divided into twelve months of irregular lengths.

Month No.	Days in Month
1, 3, 5, 7, 8, 10, 12	31
4, 6, 9, 11	30
2	28 or 29

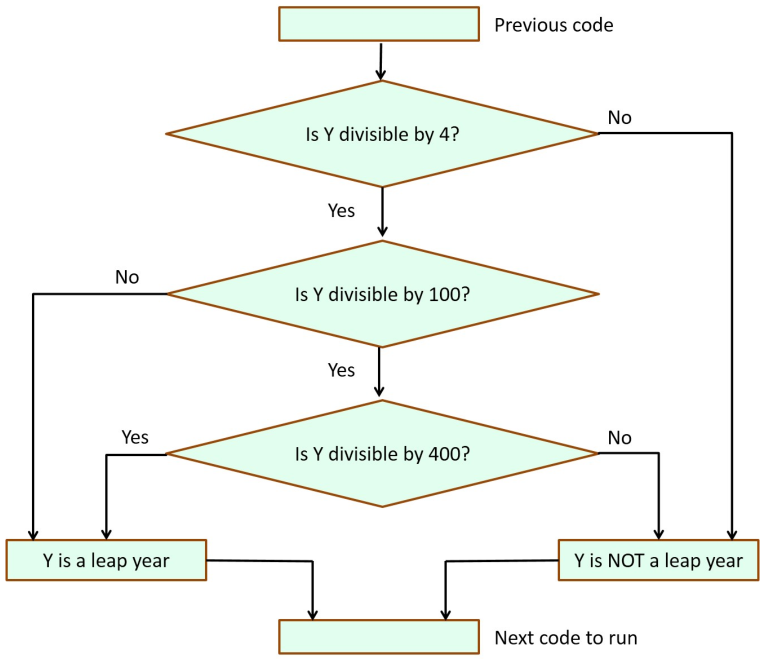
In a leap year, February has 29 days (instead of its usual 28 days). February 29 is called leap day.

Month No.	Month Name	Days in Month
1	January	31
2	February	28 or 29
3	March	31
4	April	30
5	May	31
6	June	30
7	July	31
8	August	31
9	September	30
10	October	31
11	November	30
12	December	31

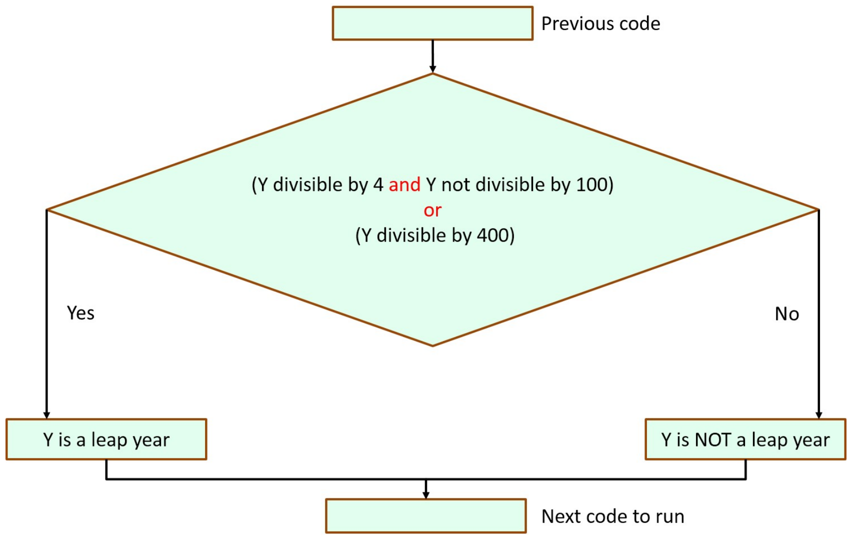
Leap Years

Expressed by the following flowchart, a leap year satisfies *one* of these two conditions:

(a) the year number is divisible by 4 but not by 100, or
(b) the year number is divisible by 400;



For example, condition (a) implies that the century years 1700, 1800, and 1900 are not leap years, and that 1796, 1804 are leap years. Condition (b) implies that the century years 1600 and 2000 are (were) leap years. Combining the conditions, the flowchart above can also be summarized as follows:



Days of the Week

Zeller's Algorithm

The day of the week for a valid Gregorian calendar date (*day*, *month*, *year*) can be calculated by using Zeller's algorithm[1,2] described below. Note that all of the arithmetic operations involved should be performed using integer operations (that is, no fractions!).

$$\begin{aligned}
w &= (14 - month)/12 && (1)\\
m &= month + 12w && (2)\\
x &= year - w && (3)\\
y &= x\%100 && (4)\\
c &= x/100 && (5)\\
d &= day && (6)\\
z &= (d + (26(m+1))/10 + y + y/4 + c/4 + 5c)\%7 && (7)
\end{aligned}$$

The result of Zeller's algorithm, z, obtained at step (7), is an integer from 0 through 6 (why?) that represents a day of the week as follows:

0: Saturday, 1: Sunday, 2: Monday, 3: Tuesday, 4: Wednesday, 5: Thursday, 6: Friday

Evaluation Criteria

Correctness of execution of your program	70%
Description of purpose of the program	5%
Documentation for nontrivial steps in code	10%
Choice of meaningful variable names	5%
Indentation and readability of program	5%
Proper use of required Java concepts	5%

[1]The Reverend Christian Zeller (1822-99). (See http://www.merlyn.demon.co.uk/zeller-c.htm)
[2]Chr. Zeller, "Kalender-Formeln", Acta Mathematica, Vol. 9, pp. 131-136. (Nov. 1886). In German.