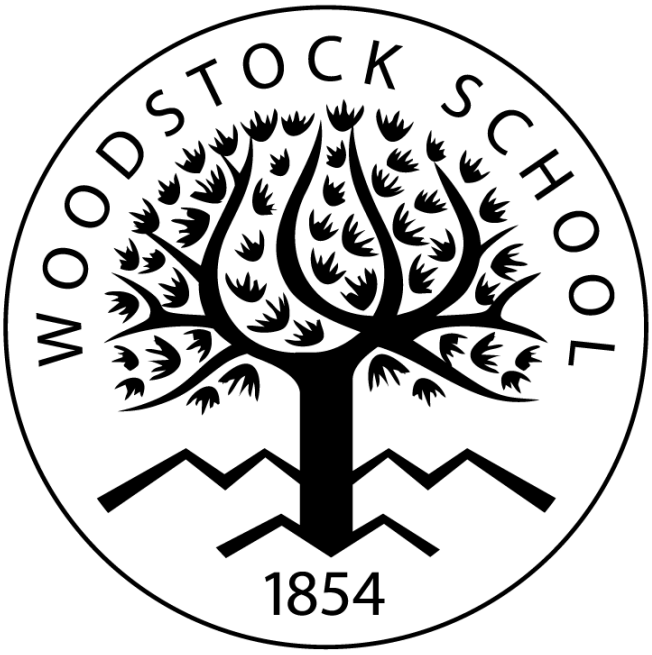
Calendar Lab

AP Computer Science A



Name: _____

Contents

Background

In this lab, you will be creating a method for finding the day of the week for any given date (day, month, year) and will use that method for generating a full calendar display for a given month.

There have been a number of different algorithms developed to calculate what day of the week a given past or future date falls on. Although many of these methods require fairly complex look-up tables, a formula has been developed that can be used to directly calculate the day of the week without having to store or process these tables. This formula is given below:

$$w = \left(d + \lfloor 2.6m - 0.2 \rfloor + y + \left\lfloor \frac{y}{4} \right\rfloor + \left\lfloor \frac{c}{4} \right\rfloor - 2c\right) \textbf{ mod } 7$$

Note the following:

- $\lfloor x \rfloor$ is the *floor* operator, accessible via `Math.floor(x)`
- $\textbf{mod}$ is the *modulus* operator (`%`)
- Y is the year minus 1 for January or February
- y is the last 2 digits of Y
- c is the first 2 digits of Y
- d is the day of the month (1 to 31)
- m is the shifted month (March=1,...,February=12)
- w is the day of the week (0=Sunday,...,6=Saturday)

Remarkably, this method will work regardless of whether or not a given year is a leap year. Consider the following examples.

Examples

July 9, 1983

$$w = \left(9 + \lfloor 2.6(5) - 0.2 \rfloor + 83 + \left\lfloor \frac{83}{4} \right\rfloor + \left\lfloor \frac{19}{4} \right\rfloor - 2(19)\right) \textbf{ mod } 7 = 6$$

Result: *Saturday*

April 15, 2016

$$w = \left(15 + \lfloor 2.6(2) - 0.2 \rfloor + 16 + \left\lfloor \frac{16}{4} \right\rfloor + \left\lfloor \frac{20}{4} \right\rfloor - 2(20)\right) \textbf{ mod } 7 = 5$$

Result: *Friday*

January 30, 2000

$$w = \left(30 + \lfloor 2.6(11) - 0.2 \rfloor + 99 + \left\lfloor \frac{99}{4} \right\rfloor + \left\lfloor \frac{11}{4} \right\rfloor - 2(19)\right) \textbf{ mod } 7 = 0$$

Result: *Sunday*

Applications

Question #1: Use the formula described in the background to calculate the day of the week for your birthday this year. Then, calculate the day of the week for your original birthday.

Question #2: Explain what happens when you attempt to calculate the day of the week for a non-existent day (February 29, 2017 or July 33, 2020). Is the result consisitent with what you would expect? Explain why or why not.

Question #3: In September, 1752, Britain and its colonies began using the Gregorian calendar. This caused Thursday, September 14, 1752 to be preceded by Wednesday, September 2, 1752. Will the formula described in the background take this calendar switch into account? Explain why or why not.

Question #4: What modification, if any, could you make to the formula described in the background to allow for the beginning of the week ($w = 0$) to be Monday?

Activity #1

Introduction

In this activity, you will be creating two methods: one that will use the formula described in the background and return the numeric value of w and one that will return the day of the week as a string. These methods will be important for the second activity.

Exercises

1. Implement the `calculateDayOfWeek()` method. This method should take as parameters the `day`, `month`, and `year` for the desired date and return the day of the week for that date as a number between 0 and 6.
2. Implement the `getDayOfWeek()` method. This method should take as parameters the `day`, `month`, and `year` for the desired date and return the day of the week as a `String` ("Monday", "Tuesday", etc.).

   Note: Your `getDayOfWeek()` method should use the `calculateDayOfWeek()` method previously implemented.

Questions

Question #5: Briefly explain the method you used to calculate y and c from the `year` variable that was passed.

Question #6: Why do you think it is considered "better practice" to use the `calculateDayOfWeek()` method within the implementation of `getDayOfWeek()` rather than just repeating the use of the formula?

Activity #2

Introduction

In this activity, you will be creating a method to display a calendar for a given month and year. The output of your method should create a calendar similar to the one shown below.

March, 2018

Mo	Tu	We	Th	Fr	Sa	Su
			01	02	03	04
05	06	07	08	09	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

In particular, note the following:

- The month and year are displayed above the main calendar.
- There are blank spaces for days prior to the start of the month and after the end of the month.
- Single digit days are displayed with a leading "0".

Exercise

1. Implement the `isLeapYear()` method which will take a year as input and return `true` if the given year is a leap year and `false` otherwise.
2. Implement the `printCalendar()` method which will take a month and year as input and produce the output described in the introduction to this activity.
   Note: Unlike the formula you used for `calculateDayOfWeek()`, your implementation of `printCalendar()` will have to use `isLeapYear()` in order to handle February's calendar output.

Questions

Question #7: Briefly explain what considerations you needed to make to take leap-years into account.

Question #8: Briefly explain how you handled the variable number of days in each month.

Final Analysis

Question #9: Which part of the implementation of either `calculateDayOfWeek()` or `getDayOfWeek()` did you find most challenging? How did you overcome this challenge?

Question #10: Which part of the implementation of `printCalendar()` did you find most challenging? How did you overcome this challenge?

Question #11: Given the knowledge and opportunity, what features would you add to your implemented methods to make them more useful and/or more versatile?

Question #12: What new programming techniques or knowledge did you learn as a result of this lab?

Template Class & Test Cases

```
/**
 * Calendar Lab (Template Class and Test Cases)
 * This is the template class and test cases for the Calendar Lab.
 * Written for the Woodstock School in Mussoorie, Uttarakhand, India.
 *
 * @author Jeffrey Santos
 * @version 1.0
 */

public class Calendar {
  public static void main(String[] args) {
    //  Tests for calculateDayOfWeek:
    System.out.println(calculateDayOfWeek(9, 7, 1983));   // Output: 6
    System.out.println(calculateDayOfWeek(15, 4, 2016));  // Output: 5
    System.out.println(calculateDayOfWeek(30, 1, 2000));  // Output: 0

    //  Tests for getDayOfWeek:
    System.out.println(getDayOfWeek(9, 7, 1983));         // Output: "Saturday"
    System.out.println(getDayOfWeek(15, 4, 2016));        // Output: "Friday"
    System.out.println(getDayOfWeek(30, 1, 2000));        // Output: "Sunday"

    //  Tests for printCalendar:
    //   Use the calendar feature of your computer to verify correct output.
    printCalendar(7, 1983);
    printCalendar(4, 2016);
    printCalendar(1, 2000);
  }

  /**
   * Calculates the day of the week as a number between 0 and 6.
   * (0=Sunday, 6=Saturday)
   *
   * @param day
   *  Precondition: day is a valid day of the given month.
   * @param month
   *  Precondition: month is a valid month of the year.
   * @param year
   *  Precondition: year is a valid year after the adoption of the Gregorian Calendar.
   * @return  A number corresponding to the day of the week for the given date.
   *          (0=Sunday, 6=Saturday)
   */
  public static int calculateDayOfWeek(int day, int month, int year) {
    //  To be implemented in Activity #1, Exercise 1
  }

  /**
   * Returns the day of the week as a string.
   *
   * @param day
   *  Precondition: day is a valid day of the given month.
   * @param month
   *  Precondition: month is a valid month of the year.
   * @param year
   *  Precondition: year is a valid year after the adoption of the Gregorian Calendar.
   * @return  A string corresponding to the day of the week for the given date.
   */
  public static String getDayOfWeek(int day, int month, int year) {
    //  To be implemented in Activity #1, Exercise 2
  }

  /**
   * Determines whether or not a given year is a leap year.
   *
```

```
    * @param year
    *  Precondition: year is a valid year after the adoption of the Gregorian Calendar.
    * @return  true if the given year is a leap year, false otherwise.
    */
   public static boolean isLeapYear(int year) {
     //  To be implemented in Activity #2, Exercise 1
   }

   /**
    * Prints a visual calendar for the given month and year.
    *
    * @param month
    *  Precondition: month is a valid month is the year.
    * @param year
    *  Precondition: year is a valid year after the adoption of the Gregorian calendar.
    */
   public static void printCalendar(int month, int year) {
     //  To be implemented in Activity #2, Exercise 2
   }

   /**
    * A fix to the Java built in modulus operator (%) to behave more
    * like mathematical modulo.
    * @param n
    * @param m
    * @return returns n mod m as expected mathematically
    */
    public static int mod(int n, int m) {
      return (n % m + m) % m;
    }
}
```

Scoring Matrix

Calendar Lab											
Applications			**Activity #1**			**Activity #2**			**Final Analysis**		
Q01	**1**		**EX1**	**5**		**EX1**	**3**		**Q09**	**1**	
Q02	**1**		**EX2**	**4**		**EX2**	**6**		**Q10**	**1**	
Q03	**1**		**Q05**	**1**		**Q07**	**1**		**Q11**	**1**	
Q04	**1**		**Q06**	**1**		**Q08**	**1**		**Q12**	**1**	

Comments: