**CSE 174 – Fall 2016**

**PROGRAM #8: 62 points – Due Sunday, October 30, by 11:59 p.m.**

**Outcomes:**

* Implement methods following specifications.

**Scoring:**

* If you do not submit a zip file containing your source code, your score will be zero.
* If you submit source code that does not compile, your score will be zero.
* If you submit source code without the correct class name (**Program8**) or the correct method names (see below), you will receive at most half-credit for this assignment.
* Deductions will be made for not meeting the usual requirements:
* Source code that is not formatted according to the usual guidelines, including commenting each method to explain its purpose. See program1 for an example of how we comment our methods.

|  |  |  |
| --- | --- | --- |
|  | **Full credit** | **Partial credit** |
| **Implement the 12 date-related methods**  **(48 points - 4 points per method)** | Your program implements each method as specified, returning correct results for all test cases. | Your program implements the method specified, but it sometimes produces incorrect output. |
| **Implement the main() method**  **(10 points)** | Your main method produces the expected results for all valid and invalid dates. | Your main() method is implemented, but with some errors. |
| **Write methods that avoid code duplication**  **(4 points)** | Each method calculates only what it is supposed to calculate, calling on other methods as needed | Some methods perform unneeded calculations that should have been handled by another method |

**Important note:**

In this assignment, your code will be tested using code written by your instructor. Your instructor will attempt to call your methods. For this reason, it is essential for you to name your class and your methods exactly as specified, *and* to make sure the parameter and return types perfectly match what is specified in this assignment. **If you do not follow these requirements, your instructor's code will not work, and you will lose at least half the credit on this assignment. If you are not sure how to write one of the methods, write a stub for the method. Otherwise your instructor's code will not compile and you will lose at least half the credit on this assignment.**

**Assignment:**

1. Create a class named **Program8**.
2. Implement the following methods.

In the first three methods below (parseMonth, parseDay, parseYear), you should assume that the String date always uses the same format, with a 1- or 2-digit integer for the month, followed by a hyphen, followed by a 1- or 2-digit integer for the day, followed by a hyphen, followed by a 4-digit integer for the year.

|  |
| --- |
| **public static int parseMonth(String date)**  Takes a date such as "03-4-2017" and returns the number corresponding to the month (in this case, 3). This method should not check if the numbers are valid. For example, getMonth("98-99-2000") should return 98. |
| **public static int parseDay(String date)**  Takes a date such as "3-4-2017" and returns the number corresponding to the day (in this case, 4). This method should not check if the numbers are valid. For example, getDay("98-99-2000") should return 99. |
| **public static int parseYear(String date)**  Takes a date such as "3-4-2017" and returns the number corresponding to the day (in this case, 2017). This method should not check if the numbers are valid. For example, getDay("98-99-2000") should return 2000. |
| **public static boolean isLeapYear(int year)**  Takes a year such as 1976 and returns true if it is a leap year, and false otherwise. Your method should work correctly for all years beginning with the year 1600. You do not need to test it with years before 1600. In your codingbat homework, you should have written a solution to this method. You could use the same logic for this method. |
| **public static int daysInMonth(boolean leap, int month)**  Takes a boolean flag indicating whether a year is a leap year, along with a month, and returns the number of days in that month. For example, daysInMonth(false, 4) should return 30, and daysInMonth(true, 2) should return 29. If the month is not in the range 1 to 12, return -1. |
| **public static boolean isValid(int month, int day, int year)**  Takes a month, day, and year, and returns true if the numbers correspond to a valid date, and false otherwise. For example, isValid(3, 14, 1916) should return true. But all of the following, for example, should return false: isValid(15, 15, 2000) and isValid(2, 29, 2100) and isValid(1, 2, -9). |
| **public static int daysRemaining(int month, int day, int year)**  Takes a month, day, and year, and returns the number of days remaining in the year, including the specified day. If the given date is not valid, returns -1. For example, daysRemaining(1, 1, 2016) should return 366, daysRemaining(12, 31, 2000) should return 1, and daysRemaining(12, 1, 1967) should return 31. |
| **public static int dayOfWeek(int month, int day, int year)**  Takes a month, day, and year, and returns an int corresponding to the day of the week (0 for Sunday, 1 for Monday, 2 for Tuesday, 3 for Wednesday, 4 for Thursday, 5 for Friday, or 6 for Saturday). For example, dayOfWeek(10, 17, 2016) should return 1. Your method should give correct values for any date in the year 1600 or beyond. If the date is not valid, your method should return the value -1. This will likely be the trickiest method to implement in this assignment. You are encouraged to research this. If you find Java code online that partially or completely solves this problem, you may use it but then you must give credit in the comments of the method. Give the exact location where you found the code you are using so that others may find the exact same code you found. The only unacceptable solution is to get code from someone else in the class. |
| **public static String getDayName(int day)**  Takes an integer for a day (0 for Sunday, 1 for Monday, 2 for Tuesday, 3 for Wednesday, 4 for Thursday, 5 for Friday, or 6 for Saturday), and returns the name of the corresponding day (Sunday, Monday, etc.). For example, getDayName(5) should return "Friday". If the number of the day is outside of the range 0 to 6, the method should return the String "ERROR". **Read about Java's switch statement in chapter 3 and use it to implement this method. Do not use any if statements in this method.** |
| **public static String getMonthName(int month)**  Takes an integer for a month, and returns the name of the corresponding month (January, February, etc.). For example, getMonthName(3) should return "March". If the number of the month is outside of the range 1 to 12, the method should return the String "ERROR". **Read about Java's switch statement in chapter 3 and use it to implement this method. Do not use any if statements in this method.** |
| **public static String dateString(int month, int day, int year)**  Takes a month, day, and year, and returns a String representation of the corresponding date . For example, dateString(10, 17, 2016) should return "Monday, October 17, 2016". If the date is not valid, this method should return "ERROR". Note that this method should do very little work on its own. Instead, it should use many of the other methods you wrote. |
| **public static String nextDateString(int month, int day, int year)**  Takes a month, day, and year, and returns a String representation of the next corresponding date . For example, nextDateString(10, 31, 2016) should return "Tuesday, November 1, 2016". If the specified date is not valid, this method should return "ERROR". Note that this method should do very little work on its own. Instead, it should use many of the other methods you wrote. |
| **public static void main(String[] args)**  Implement this method to ask the user for a date, and display information about that date. This method should not do any calculations but, rather, should call on the other methods you have written. The input and output should be formatted as shown in the sample runs below. |

**Sample runs:**

How the program should work when the user enters a valid date that is a leap year:

|  |
| --- |
| Enter a date: **10-17-2016**  Monday, October 17, 2016  Days in October, 2016: 31  Days remaining in 2016: 76  Leap year: yes  Next day: Tuesday, October 18, 2016 |

How the program should work when the user enters a valid date that is not a leap year:

|  |
| --- |
| Enter a date: **12-31-2100**  Friday, December 31, 2100  Days in December, 2100: 31  Days remaining in 2100: 1  Leap year: no  Next day: Saturday, January 1, 2101 |

How the program should work when the user enters a date that is not valid:

|  |
| --- |
| Enter a date: **2-29-1915**  This is not a valid date. |

**Additional requirements:**

1. zip file requirement: In a folder named program8, put exactly one file: your source code named Program8.java. Zip the folder named program8, and submit that zip file.
2. Java has a variety of classes designed for date calculations. You should not use these for this assignment. Solutions that use any of Java's Date classes or methods will at most receive half credit.
3. Each method must be commented to explain its purpose. Comments should explain what the method does, along with any "special" cases. For example, if your method is supposed to return "ERROR" for certain values, explain that in the comments.
4. **If you are not sure how to write one of the methods, you must write a stub for the method. Otherwise your instructor's code will not compile and you will lose at least half the credit on this assignment.**
5. Avoid repeating calculations in your code. For example, there should be only ONE place in your code that calculates whether a date is valid. All other methods that need to check if a date is valid should call that one method.

**Advice:**

1. You are allowed to write additional "helper methods" if you want.
2. Focus on the little methods. Find the methods that seem easiest to write, and write them first. You do not need to write these in the order that they appear in the instructions. For example, by now you should find it easy to write the isLeapYear method. So, you could start there.
3. Each time you write a new method, test it either in Dr. Java's Interactions Pane or in your main() method.
4. Do not worry about the user input and formatting until you have written most of the smaller methods. You should not even bother with that part of the program until the very end.
5. Double check: Have you named your class and your required methods as specified? Are the parameter and return types correct? If any of these are incorrect, your instructor's test code will fail, and you will lose at least half credit on this assignment. If you are not sure how to write one of the methods, write a **stub** for the method. Otherwise your instructor's code will not compile and you will lose at least half the credit on this assignment.