1. **(10 points) Chapter 4: Self-check problems #2, #5, #12, #15, and #27.**

   If you use 2nd edition: Self-check problems #2, #3, #9, #12, #17 and #24.

   For your benefit, I suggest you complete all of the self-check problems on your own. However, I will just collect the answers to the above six problems. Please type the answers into a document.

   a. NO NEED to copy original questions, just put down the question number and the answer.
   b. Name the file as "***LastnameFirstname4.doc***" (or ".docx" or ".pdf", where "Lastname" is your last name, and "Firstname" is your first name), submit it online.

2. **(10 points) Chapter 4: Exercise 18 "Word Count"**

   If you use 2nd edition: Chapter 4 Exercise 15 "Word Count"

   a. Download the attached "`WordCount.java`" file.
   b. Implement the method `wordCount` as described in the problem.
      i. Please note: This is NOT an interactive program, therefore, DO NOT use `scanner`.
      ii. The method `wordCount` should accept a `String` as its parameter and return the number of words in the `String`. A word is a sequence of one or more nonspace characters (any character other than whitespace ' ').
      iii. Some examples of expected output:

      The call `wordCount("hello")` should return 1
      The call `wordCount("how are you?")` should return 3
      The call `wordCount("  how are you?")` should return 3
      The call `wordCount("how are you?  ")` should return 3
      The call `wordCount("how are you ?")` should return 4
      The call `wordCount("I am 3 years old")` should return 5
      The call `wordCount("3 x 4 = 12")` should return 5
      The call `wordCount("Characters !@#$%^&*")` should return 2
      The call `wordCount("  ")` should return 0

   c. Submit the final "**`WordCount.java`**" file **(DO NOT change the file name)** online.

3. **(30 points) Chapter 4: Programming projects 4 "Birthdays"**

   a. Download the attached "`Birthdays.java`" file.
   b. Modify the code so that the program prompts the user three questions and output six lines of messages as described below:

Your program asks for today's date and two users' birthdays and prints information about them. For both birthdays, the program prints the absolute day of the year on which that birthday falls, the number of days until the user's next birthday. Next the program shows which user's birthday comes sooner in the future. If it is a user's birthday today or if the two birthdays are the same, different messages are printed. Four sample dialogues are shown below, where user input is underlined and expected output is bold. The input date format: *Month Day* (with a single space in between).

```
Please enter today's date (month day): 1 29
Today is 1/29/2016, day #29 of the year.

Please enter person #1's birthday (month day): 10 17
10/17/2016 falls on day #291 of 366.
Your next birthday is in 262 day(s).

Please enter person #2's birthday (month day): 11 23
11/23/2016 falls on day #328 of 366.
Your next birthday is in 299 day(s).

Person #1's birthday is sooner.

Did ye know? 9/19 be International Talk like a Pirate day.
Arr, me mateys, arr!
============================================================

Please enter today's date (month day): 2 14
Today is 2/14/2016, day #45 of the year.

Please enter person #1's birthday (month day): 1 10
1/10/2016 falls on day #10 of 366.
Your next birthday is in 331 day(s).

Please enter person #2's birthday (month day): 2 28
2/28/2016 falls on day #59 of 366.
Your next birthday is in 14 day(s).

Person #2's birthday is sooner.

Did ye know? 9/19 be International Talk like a Pirate day.
Arr, me mateys, arr!
============================================================

Please enter today's date (month day): 1 1
Today is 1/1/2016, day #1 of the year.

Please enter person #1's birthday (month day): 1 1
1/1/2016 falls on day #1 of 366.
Happy Birthday! :)

Please enter person #2's birthday (month day): 12 31
12/31/2016 falls on day #366 of 366.
```

```
Your next birthday is in 365 day(s).

Person #1's birthday is sooner.

Did ye know? 9/19 be International Talk like a Pirate day.
Arr, me mateys, arr!

============================================================

Please enter today's date (month day): 2 28
Today is 2/28/2016, day #59 of the year.

Please enter person #1's birthday (month day): 2 29
2/29/2016 falls on day #60 of 366.
Your next birthday is in 1 day(s).

Please enter person #2's birthday (month day): 2 29
2/29/2016 falls on day #60 of 366.
Your next birthday is in 1 day(s).

Wow, you share the same birthday! ^_^

Did ye know? 9/19 be International Talk like a Pirate day.
Arr, me mateys, arr!
```

Note that the prompt messages and outputs need to be in the **exact format** as shown above. Lastly, the program prints a fun fact about your own birthday. Try searching Wikipedia, Bing, or Google for interesting facts about your date of birthday. For example, if yours is Sep. 19, you could print the following fact at the end:

```
Did ye know? 9/19 be International Talk like a Pirate day.
Arr, me mateys, arr!
```

Please note: this fun fact should appear for EVERY run, but NOT appear only for a specific date (i.e., your birth date).

c. Keep in mind, this program involves date comparisons in **2016**, which is a "**leap year**." Leap years have an extra 366th day, which occurs as **February 29th.**
d. Since this is an interactive program, it behaves differently when given different input. **The examples above may not show all possible cases. Please examine all example outputs shown above and do your own testing.**
e. For this assignment you are **limited to the language features in Chapters 1-4** of the textbook. In particular, Do NOT use ANY library function to count the absolute day of the year, e.g. `GregorianCalendar` class or `java.util.Calendar`. (If you don't know what it is, just ignore it ☺).
f. **Development strategy**: write your code **in stages**, continually making small improvements.

i. One major task in this program is computing the **absolute day of the year** on which each user's birthday falls in 2016. Jan 1 is day #1. Jan 2 is #2. Jan 31 is #31. Feb 1 is #32. And so on, up to Dec 31, which is #366.
ii. You must compute absolute days of the year by writing a particular static method. This method should accept parameters representing a month and day and should return the absolute day of the year 2016 that is represented by those parameters. For example, calling this method with the parameter values of 2 and 13 (representing February 13th, 2016) should return 44, and calling it with parameter values of 9 and 19 (representing September 19th, 2016) should return 263. This method should not produce any console output, though the result it returns can be printed by code elsewhere in your program.
iii. Your method must compute its value using a cumulative sum as described in Chapter 4 of the textbook. To perform this calculation, you will also need code to evaluate the number of days in each particular month. You might put this code into a method that accepts a month parameter and returns the number of days in that month.
iv. Another major task is computing how many days remain until each user's next birthday. There are several cases to consider. If the user's birthday falls after today's date, it is within the year 2016. However, if it falls before today's date, the next birthday occurs in the year 2017, so you'll need to think of a way to count the days that "wrap around" between today and the end of 2016, and then the start of 2017 to the user's birthday in 2017.

g. Remember, your program will be graded both on "external correctness" (whether your program compiles and produces exactly the expected outputs), and "internal design and style" (whether your source code follows the **style guide**).

h. Submit the final "`Birthdays.java`" file **(DO NOT change the file name)** online.

Note 1: In the instructional examples above, in order to distinguish between user input and expected output, user input is underlined and expected output is **bold**. You DO NOT need to display this format. ☺

Note 2: To parse the input dates (*Month Day* with a single space in between), the easiest way is to treat them as **two separate integers**. See ***Chapter 3 Lecture Notes slide #53*** for example.