CSE 190 D, Winter 2013
Programming Assignment #6: Weather (20 points)
Due Monday, February 18th, 11:30 PM

This assignment focuses on arrays and file/text processing. Turn in a file named `Weather.java`. You will also need the input file `weather.txt` from the course web site. Save this file in the same folder as your program.

In this assignment you read an input file containing average daily temperatures and produce information about average monthly and yearly temperatures. The file is guaranteed to contain exactly 12 lines of input, one for each month in the year. The lines each start with an `int` representing the number of days in the month. The `int` is followed by `double`s representing the average temperature for each day in the month.

Input file `weather.txt` (partial):

```
31 44 43.5 43 43 43 43 45 45 45 44 45 45 45 46 46 45 45 46 46 46 45 45 45 45 45 44 45 45 45 45 46
29 47 46 48 49 48 49 49 49 48 49 49 50 48 48 48 49 49 49 49 50 50 50 50 50 50 49 51 51 51
31 49 50 48 49 50 51 51 52 52.2 52 52 51 52 52 52 52 53 53 54 54 54 54 54 54 53 53 53 54 55 55 54
30 54 54 56 56 56 56 56 56 57 57 57 57 57 57 58 58 56 57 57 58 58 58 58 58 59 60 60 60 61 62.1
```
...

Program Behavior:

Your program outputs results to a file called `temperatures.txt`. The results include the average temperature seen each month and the number of days in each month that were above average temperature. At the end of the results it includes the average temperature for the year and the number of months with an above yearly average monthly average.

Log of execution:

```
This program has output weather averages to temperatures.txt
```

Output file `temperatures.txt` after above execution:

```
Month 1 average temperature: 44.774193548387096
23 days were above average temperature

Month 2 average temperature: 49.06896551724138
10 days were above average temperature

Month 3 average temperature: 52.32258064516129
15 days were above average temperature

Month 4 average temperature: 57.46666666666667
13 days were above average temperature

Month 5 average temperature: 64.16129032258064
16 days were above average temperature

Month 6 average temperature: 69.5
14 days were above average temperature

Month 7 average temperature: 75.12903225806451
16 days were above average temperature

Month 8 average temperature: 74.80645161290323
17 days were above average temperature

Month 9 average temperature: 69.36666666666666
14 days were above average temperature

Month 10 average temperature: 59.483870967741936
15 days were above average temperature

Month 11 average temperature: 50.43333333333333
14 days were above average temperature

Month 12 average temperature: 45.54838709677419
15 days were above average temperature

Average temperature for the year: 59.338453219626736
6 months were above average temperature
```

Implementation Guidelines, Hints, and Development Strategy:

The main purpose of this assignment is to demonstrate your understanding of arrays and array traversals with `for` loops. Therefore, you should use arrays to store data. In particular, **your average daily temperatures and average monthly temperatures should all be stored using arrays**. You may not read any part of your file more than once.

Recall that you can print any array using the method `Arrays.toString`. For example:

```
int[] numbers = {10, 20, 30, 40};
System.out.println("my data is " + Arrays.toString(numbers));      // my data is [10, 20, 30, 40]
```

We suggest that you start this program by writing the code to read the input file. Try writing code to simply read each line of temperatures and print them. Read each line from the input file using `Scanner`'s `nextLine` method. This will read an entire line of input and return it as a `String`.

Next, write code to read through a line of temperatures and store them in an array. Remember the first number in the row is the number of days in the month and so also the number of temperatures you need to store.

Once you have the numbers stored in arrays properly, compute their average and then use it to count the number of temperatures above average.

When your program is working correctly for each month, add in the code to keep track of the monthly average temperatures and find their average. You are guaranteed that your file will contain 12 months (12 lines).

We also suggest that you first get your program working correctly printing its output to the *console* before you save the output to a file. Once you have your program printing correct output to the console, save the output to a file by using a `PrintStream` as described in last week's lecture and Section 6.4 of the textbook.

You may assume that the input file exists, is readable, and contains valid input. Your program should overwrite any existing data in the output file (this is the default `PrintStream` behavior).

Style Guidelines:

For this assignment you are required to have a **class constant** for the number of months in the file. For full credit it should be possible to change this constant value and cause your program to change its behavior. You may use additional constants if they make your code clearer.

We will grade your method structure strictly on this assignment. Use at least **three nontrivial methods** besides `main`. These methods should use parameters and returns, including arrays, as appropriate. The methods should be well-structured and avoid redundancy. No one method should do too large a share of the overall task. The textbook's case study at the end of Chapter 7 is a good example of a larger program with methods that pass arrays as parameters.

Your `main` method should be a concise summary of the overall program. It is okay for `main` to contain some code such as `println` statements. But `main` should not perform too large a share of the overall work itself, such as looping over the temperatures. Also avoid "chaining," when many methods call each other without ever returning to `main`. For reference, our solution is around 70 lines long and has 3 methods besides `main`, though you don't need to match this.

We will also check strictly for redundancy on this assignment. If you have a very similar piece of code that is repeated several times in your program, eliminate the redundancy such as by creating a method, by using `for` loops over the elements of arrays, and/or by factoring `if/else` code as described in section 4.3 of the textbook.

Since arrays are a key component of this assignment, part of your grade comes from using arrays properly. For example, you should reduce redundancy as appropriate by using **traversals** over arrays (`for` loops over the array's elements). This is preferable to writing out a separate statement for each array element (a statement for element `[0]`, then another for `[1]`, then for `[2]`, etc.). Also carefully consider how arrays should be passed as parameters and/or returned from methods as you are decomposing your program. Recall that arrays use *reference semantics* when passed as parameters, meaning that an array passed to a method can be modified by that method and the changes will be seen by the caller.

You are limited to features in Chapters 1 through 7. Follow past style guidelines such as indentation, names, variables, types, line lengths, and comments (at the beginning of your program, on each method, and on complex sections of code).