Problem Set 3

The following problem set will be worth 100 points. The code will be submitted electronically via Canvas using the "Problem Set 3" dropbox. The assignment is **due at the start of the class two weeks from the date it was assigned**.

Your code will be graded on both elegance and *user-friendliness*.

Exercise #1 – Average Problem (10pts)

Write a program which calculates the average of *N* integers. The program should prompt the user to enter the value of *N*. Afterwards the user must enter all *N* of the integers. If the user enters a non-positive value for any of the integers, an exception should be thrown (and caught) with the message "N must be positive."

If there is any exception as the user is entering the *N* numbers, an error message should be displayed and the user prompted to enter the number again.

Save your solution in **Average.java**.

Exercise #2 – Formatting Problem (10pts)

Write a program which converts dates in the format:

- 12/25/2000

To the format:

- December 25, 2000

Define three custom exception classes: **DayException**, **MonthException** and **YearException**. If the user enters anything that is an invalid month (outside the range of 1 through 12), your program should throw and catch a **MonthException** and ask the user to reenter the month. If the users anything outside the range of 1 to 31 for the day, throw and catch a **DayException** and ask the user to reenter the day. If the user enters a year outside the range of 1000 to 3000, throw and catch a **YearException** and ask the user to reenter the year.

Save your solution in **DateFormatter.java**, **DayException.java**, **MonthException.java**, and **YearException.java**.

Exercise #3 – Bowling Problem (20pts)

Consider a frame of bowling pins, where each * represents a pin:

*

```
 **
 ***
 ****
 *****
```

There are five rows, and a total of 15 pins.

If we had only the top four rows, there would be 10 pins.

If we had only the top three rows, there would be a total of 6 pins.

If we had only the top two rows, there would be a total of 3 pins.

If we had only the top row, there would be one pin.

Write a recursive function that takes as input the number of rows *n* and outputs the total number of pins that would exist in a pyramid with *n* rows. Your program should allow for values of *n* that are larger than 5.

Save your solution in **Bowling.java**.

Exercise #4 – Array Problem (20pts)

Given the definition of a 2D array such as follows:

```
String[][] data = {
    {"A", "B"},
    {"1", "2"},
    {"XX","YY","ZZ"}
};
```

Write a recursive program that outputs all combinations of each subarray in order. In the previous example, the desired output might look like the following:

```
A 1 XX
A 1 YY
A 1 ZZ
A 2 XX
A 2 YY
A 2 ZZ
B 1 XX
B 1 YY
B 1 ZZ
B 2 XX
B 2 YY
B 2 ZZ
```

Your program should work with arbitrarily sized arrays in either dimension. For instance, consider the following input array:

```
String[][] data = {
    {"A"},
    {"1"},
    {"2"},
    {"XX","YY"}
};
```

Should output:

```
A 1 2 YY
A 1 2 YY
```

Save your solution in **RecursiveArray.java**.

Exercise #5 – Substring Problem (20pts)

Write a recursive method with the following signature:

```
public static boolean contains(String haystack, String needle)
```

The method should return **true** if the needle is contained inside the haystack and **false** if otherwise.

For instance consider the following:

```
contains("Java programming", "ogr") should return true
contains("Java programming", "grammy") should return false
```

Do not use the substring method in java.lang.String in your solution. Save your solution in **StringContains.java**.

Exercise #6 – Dictionary Problem (20pts)

The word ladder game was invented by Lewis Carroll in 1877. The idea is to being with a start word and then change one letter at a time until you arrive at the end word. Each word along the way must be an English word.

For example, starting from FISH, you can make the following word ladder to MAST:

FISH, WISH, WASH, MASH, MAST

Write a recursive program to find the word ladder given a start word and an end word, or determines whether no word ladder exists. Use the *words.txt* file (provided) as your dictionary of valid words. You program doesn't need to find the shortest word ladder, any ladder will work if one exists.

Submission Requirements: Submit the aforementioned files in a zip file with the naming strategy:

First initial + last name + PS + problem set number.zip

As an example, I would submit the code in a zip file named **mmeluskyPS3.zip**. Submit your zip file via the "Problem Set 3" Canvas dropbox before the date of the close of the assignment.\