Intro Level:

Problem #: Computing Means

Given a series of numbers ending with a 0, compute and write their arithmetic (AM), geometric (GM) and harmonic (HM) means, excluding the 0 flag value at the end of the input line. Your program should assume that all the numbers in the series are positive double numbers.

The program should continue asking the user for lists until the user enters a 0 as the first number in the list.

$$AM = \frac{1}{n} \sum_{i=1}^{n} x_i \quad GM = \sqrt[n]{\prod_{i=1}^{n} x_i} \quad HM = \frac{n}{\sum_{i=1}^{n} 1/x_i}$$

Sample Conversation:

Enter a list of positive numbers ending in 0, 0 to quit:
3 4 5 6 7 9 0
Arithmetic mean: 5.666666666666667
Geometric mean: 5.320353736538501
Harmonic mean: 4.983520105471325
Enter a list of positive numbers ending in 0, 0 to quit:

Problem #: Computing a Weighted Average

The program should take the first number to be the number of weights (up to 50), the rest of the line to the be the weights expressed as doubles, then accept lines with the same number of values, computing and writing the weighted average of those values, until the user enters a 0 as the first value on a line.

You may use double arrays or the java.util.ArrayList class.

Note that the sum of weights might not add up to 1.

Hint: Use the java.util.Scanner class for input, and the nextLine() method.

Sample Conversation:

The number of weights and the weights:
3 5 8 12
Enter 3 numbers, 0 to quit:
1 1 1
result: 1.0
Enter 3 numbers, 0 to quit:
88 75 95
result: 87.2
Enter 3 numbers, 0 to quit: 0
Advanced Level:

Problem #: Class Defining Weights

Write a class Weight that defines weights that are stored as integer pounds and ounces. That class should have a constructor that accepts those values to define a weight and a second public method that adds one weight to another. The signature of that method should be the following:

public void addTo(Weight otherWeight)

After the addition is performed, the weight should be normalized, if necessary, to ensure the number of ounces is less than 16. Finally the class should contain a toString method that converts a Weight object into a string.

A second class should contain the main method. It should prompt the user to enter two weights. It should compute their sum using the addTo method and display the sum as a string.

The following is a sample run of this program:

Enter first weight: 2 13
Enter second weight: 4 8
The sum of the weights is 7 lbs 5 oz

Problem #: Postfix Expression Evaluation

Write a stack class implemented with an array that has the usual operations to push, pop and check whether the stack is empty. Write a second class containing a main method that asks the user to input a postfix expression containing integers. The main method should evaluate the expression using the stack. The algorithm is to push the operands onto the stack when they are encountered and pop two whether an operator is next, perform the operation and push the result. The only allowed operators are +, - and *. You may assume that the expression is syntactically correct.

The following is a sample run of this program:

Enter a postfix expression: 2 7 + 5 1 - *
The value of the expression is 36