Due date: 10pm, Tuesday, November 15th

Goals
- to declare and use arrays to remember information that must be handled more than once

Overview
For this activity, write a program that will read a series of real values representing the yearly growth of pine trees, average the values, and then print the values and their difference from the average. The program must use an array of doubles to hold the values.

The input will contain a series of up to 10 real (double) values. The program must read until end-of-file or 10 values have been read. Do not prompt for input or echo the values as they are read. Define a constant, i.e. a public static final int, for the maximum number of values (10).

Output Specification
The first output is a count of how many values are included in the calculations. The second output is the average. Then, for each value, print the tree's index in the array, its growth value, and that value's difference from the average. Print each real value with one digit after the decimal point. An output line looks like:

Tree 3 grew 12.4 which is -2.3 from the average

If there are no values, print a message. Do not divide by zero, do not print any data, and do not compute or print the average. Just print a message and stop.

If the user inputs more than the allowed number of values, ignore those extra values. The program may print an alert message, but it should average and report on the first allowed number of values.

Approaching the Problem
Develop your solution incrementally. This problem naturally uses methods extensively. Remember that each method should have a short comment describing what it does. Writing that comment before writing the code helps you think about what you are writing and why. We suggest the following iterations.

- Declare an array and write a for loop to fill each element with its index (that is, trees[t] = t). Remember that you can use the length attribute of the array to know how many elements are allocated for the array. Write a print method to print the values. The method will also need to know how many values in the array contain useful data. That is an extra parameter. Save that, compile it, and test it.
- Use a method to fill the array with values read from the user. You also need to pass back the number of values read. Print the count in main.
- Compute and print the average. A method is good for computing the average. Be sure to catch the answer in main and print it.
- For each element in the array, print the difference from the average. You should modify your earlier print method.
- Add the error conditions. Be sure to test with no data, one value, exactly the maximum number of values, and one more than the maximum number. You may note some rounding in the difference from the average.

Sample Input
6.5 7.8 -1.4 9 22 -2.7 0 17

Sample Output
8 values included in calculations
The average is 7.3
Tree 0 grew 6.5 which is -0.8 from the average
Tree 1 grew 7.8 which is 0.5 from the average
Tree 2 grew -1.4 which is -8.7 from the average
Tree 3 grew 9.0 which is 1.7 from the average
Tree 4 grew 22.0 which is 14.7 from the average
Tree 5 grew -2.7 which is -10.0 from the average
Tree 6 grew 0.0 which is -7.3 from the average
Tree 7 grew 17.0 which is 9.7 from the average