This lab is a review of arrays. You will only have one file for this lab, the file that contains main(). You will not write any of your own classes, but you will use Java library classes.

**Decide what should be written in main() and what should be written as a static method.** Use the rules and guidelines for functions to help you with this determination. Since this is a review of arrays, complete this lab using arrays (not arrayLists).

1. Use a random number generator to populate an array of type int and max size 50. Please call this array randArray.

* The array will be of size 50, but it will probably only be partially filled.

Declare a constant variable to hold the size of the array. Then use the constant in your array declaration. Change the example code below to use variable names and values that are appropriate for this lab.

private static final int **CAPACITY** = 2;

private int[] items = new int[**CAPACITY**];

***CONSTANTS ARE IN ALL CAPS***

* Use a random number generator to determine the number of elements to be populated. For example, you may only populate 34 elements.

In the case of a partially filled array, the number of elements used is stored in a variable with a useful name like **numUsed.** Please use this name to store the value for the number of elements to be populated.

Generate random numbers using Java’s Random class.

You will need to do a bit of research into using Random. Here are some ideas to get you started.

Here is the import statement for Random:

import java.util.Random

Here is a call to the Random constructor and an assignment of the Random object that is returned.

Random rand = new Random()

**The above example uses the default Random() constructor. However, for this lab, use the one-parameter constructor whose input is a seed. Be prepared to explain for check-off the purpose of a seed in random number generation.**

Would it make sense to allow the random number generator to populate 60 elements in a 50 element array? To solve that problem, you will need to incorporate an upper bound, which is also the size of the array, **50**. **(highest number for population)**

Would it make sense to populate 0 elements in the array? Yes. But not for this lab. Also set a lower bound of perhaps 5. The program will populate at least 5 elements each time it runs. **NEEDS AT LEAST 5**

Would it make sense to populate -5 elements of the array? Or any other negative number? Make sure your random number generator only produces non-negative integers. **WHILE LOOP?**

* Use a random number generator again to populate the number of elements determined above. Example, populate 34 elements. This time your numbers should include positive, negative, and zero integers. **USE MAX ARRAY**

1. Use a second int array to keep a tally of the number of unique values in randArray. Please call this second array tally. Populate tally with zeros.

Big hint: consider using the tally array’s index to accumulate the count of values from randArray.

tally[randArray[i]]++;

This code will be in the body of a for-loop.

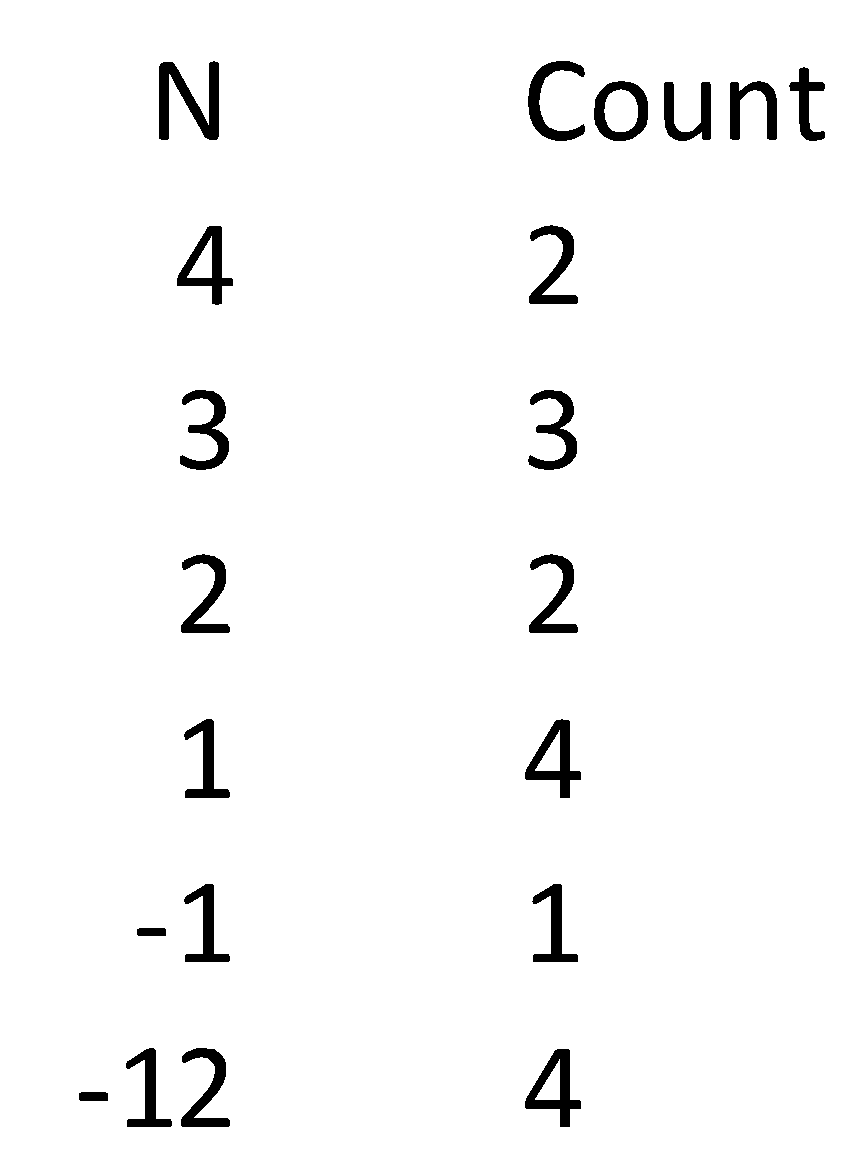
1. The output from this program will be a two-column list. The first column is a list of distinct array elements in descending order. The second list is the count of occurrences of each element.

Please get randArray to populate properly first and then move onto the tally part.

For the array

-12 3 -12 4 1 1 -12 1 -1 1 2 3 4 2 3 -12

the output would be



You will need to consider formatting options to make your lists look similar to the example. **printf**

Testing:

Be sure to test your program with various maximum values that can be stored in the array. For example, -100 to 100 might be too many possibilies, -5 to 5 might be too few.

It might be useful to hard-code numUsed in the beginning. Get randArray to populate properly first, and then randomly chose numUsed.

**DECLARE ARRAY**

**PICK NUMBER FOR NUM USED**

**BUILD ARRAY**

**use printf to print as column**