CST 338 Homework 2

Part 1 (10 points)

Write a Java program called **Histogram.java** that reads whole numbers from a user. Your program should display a list of distinct numbers in the input and the occurrence of each distinct value. After that, your program should draw a vertical bar for the occurrences using an asterisk. In the program, you can assume that the number of input values from a user is less than or equal to 30. Additionally, you can assume that the input number itself is between 0 and 9.

A sample run of your program should be like below:

```
How many input values [max: 30]?
5


Enter 5 numbers.
2
1
2
0
2


Number    Occurrence
0         1
1         1
2         3

========= Vertical Bar ========
| 3 |      *
| 2 |      *
| 1 |  * * *
===============================
| No | 0 1 2 3 4 5 6 7 8 9
===============================
```

This is another sample run of your program:

```
How many input values [max: 30]?
8

Enter 8 numbers.
5
1
5
3
9
9
3
5

Number     Occurrence
1          1
3          2
5          3
9          2

========= Vertical Bar ========
│ 3 │            *
│ 2 │        *   *       *
│ 1 │    *   *   *       *
===============================
│ No │ 0 1 2 3 4 5 6 7 8 9
===============================
```

In the assignment, your program should display **your result as the sample run**. For instance, your program should display the **distinct numbers in ascending order**. And also, the height of the vertical bar should be the same as the maximum value of “Occurrence”.

Part 2 (10 points)

The positive square root of a number S is a number $y > 0$ where $y^2 = S$. An ancient algorithm to approximate the positive square root of a number S works as follows:

Start with an initial guess, often $x_0 = \frac{S}{2}$, of the square root. Next, the algorithm improves the guess using the formula:

$$x_1 = \frac{x_0 + \frac{S}{x_0}}{2}$$

It then continues to improve the approximation with:

$$x_2 = \frac{x_1 + \frac{S}{x_1}}{2}$$

As the iterations increase, the approximations will begin to converge. For our purposes, we will decide our approximation is good enough when the difference is less than 1%.

Using this algorithm, write a Java program to approximate the positive square root of a whole number provided by the user. If the user does not provide a whole number, the program should continue and prompt the user to provide a whole number. To create the initial guess, divide the user-provided number by two. Once two consecutive approximations differ by less than 1%, your program is done. Return the approximation to only two digits after the decimal point.

Here is a sample run:

```
This program estimates square roots.
Please enter a whole number:
twenty five

Please enter a whole number (no words, just numbers):
25

Current guess: 7.25
Current guess: 5.349137931034482
Current guess: 5.011394106532552
Current guess: 5.000012953048684
```

```
The estimated square root of 25 is 5.00
```

Note: The above program terminated because

$$\frac{5.011394106532552 - 5.000012953048684}{5.011394106532552} < .01$$

When you write your programs, you should provide the following **four items at the beginning of each program**:

1. Title: File Name
2. Abstract: Overall purpose (or functionality) of the program.
3. Author: Your name
4. Date: The date you wrote the program

Points will be deducted if the above documentation is missing.