**Lab: Stepwise Refinement**

**Objectives:** This lab will give you practice in applying stepwise refinement to develop a new algorithm. If you are working in the lab, you can work in groups of 2 people.

**Background:** Stepwise refinement is a basic technique for low-level design. This refinement approach is a discipline of taking small, easily defended steps from a very generic view of an algorithm, adding a few details at each step, until the path to an actual program becomes clear.

**Procedure:**

1. Start with a general statement of the problem.
2. Pick any statement whose translation into code is non-obvious.
3. Expand that statement into 2 or more steps.
4. Repeat steps 2-4 until all statements can be easily coded.

The essential element here is that you: Expand one statement at a time replacing it by two or more, more detailed, statements and the replacements must, together, account for the whole statement being replaced.

**General Instructions:**

1. Read the Skewed Data problem description below and apply the technique of stepwise refinement to design an algorithm for the skewed data program.

Use a text editor (e.g., emacs, the Code::Blocks editor or NotePad) to make your first pseudo-code statement of the problem.

Each time you refine a step, copy and paste your current version of the function design to the end of your text file, then make your change. Try to follow the instructor’s convention of using different number of stars(\*) to indicate the level of expansion responsible for each statement. When you are done with the design, you should have a complete record of the design process you followed to reach your final design.

2. Implement your design. Your C++ code should include your pseudo-code design as comments, with the code implementing each pseudo-code statement following immediately after that statement. Your function should reside in the single file skew.cpp. You will find a driver (main) and associated files for this lab posted on Blackboard.

**Problem Description: Skewed Data**

In almost any kind of experiment where sampling is done, experimental measures are subject to random error and other sources of variation. In many cases, a histogram of the collected data would approximate a bell-shaped curve like the one shown here.

Suppose we have collected a number of data values d1, d2, …, dn. The peak of this curve will occur at the mean average of the data .

The width of the curve is controlled by the standard deviation of the data, s:

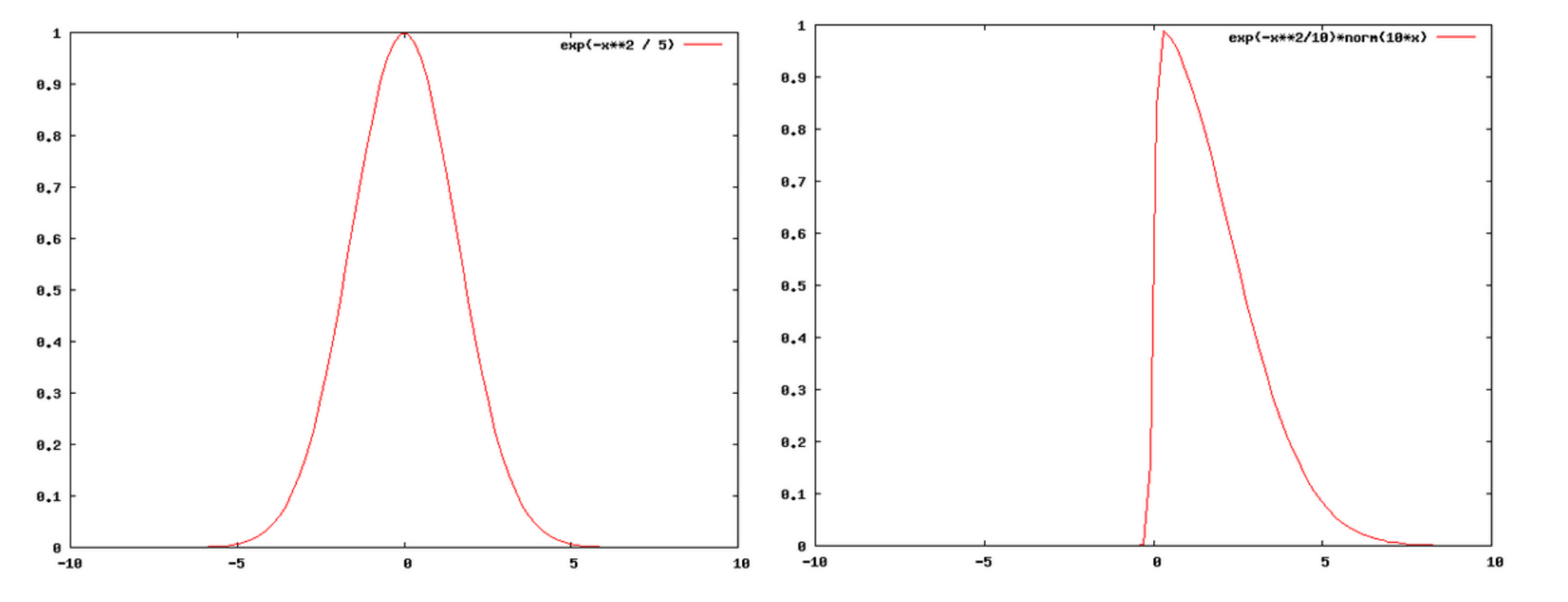
If, however, some sort of systematic bias creeps into the measurement process, then the data may be *skewed*. Experimenters are often interested in detecting skew, so that they can then track down and maybe eliminate the bias responsible for it.

One way to measure skew is:

Where Md is the median of the di (defined below). If this quantity exceeds plus or minus 1, we will say that the data is “badly skewed”.

The *median* of a collection of measurements is obtained by sorting the measurements into ascending order, then selecting the middle value from the sorted list if there are an odd number of measurements, or the midpoint between the two middle values if the number of measurements is even.

Design a program to determine whether data sets are badly skewed.



**Input**

Input consists of one or more floating point numbers, one per line. These continue to the end of input.

There will be at most 10,000 numbers in the dataset.

**Output**

The program should print one line of output. This line will consist of the phrase “Badly skewed” or “Not badly skewed”, as appropriate.

**Example**

Given the input

1.0

2.0

3.0

the output should be

Not badly skewed

Given the input

2.0

30.0

1.0

the output should be

Badly skewed