ECE 150 CODING LAB 2: STATISTICS

SUBMISSION DUE *at the end of the Lab Period*

Quidquid Latine dictum sit, altum viditu

Basic statistics and `while()` loops

You are required to write a program, `statistics`, that accepts several (one or more) command-line arguments that will be floating-point numbers representing a series of voltage readings. Valid voltage readings will be positive numbers less than 150. The list of readings will be terminated by a negative number.

Your program should compute the minimum, maximum, and average of the *valid* voltage readings.

The program will be executed as follows:

```
./Statistics <num_1> <num_2> ... <num_k> -1
```

where "`<num1>`", "`<num2>`", ... will be replaced with floating-point values. You may assume that the arguments will be valid floating-point numbers within the range `[-FLT_MAX, FLT_MAX]` and you may use "`atof`" (ASCII-to-float, requires `<stdlib.h>`) to convert the arguments.

The output should be of the form:

```
Number of voltage readings: ...
Minimum voltage: ...
Average voltage: ...
Maximum voltage: ...
```

If any voltage readings are invalid (not a positive number less than 150), then you should output, to `cerr`, the warning message:

```
Warning: invalid voltage reading <...> ignored in calculation
```

where `<...>` is whatever the invalid value was. If there are any invalid voltage readings you return a positive integer when your program completes.

If there are any errors in the input data set (think about what might be in error; there is at least one problem that can occur), you should output, to `cerr`, the message:

```
Error: Unable to compute statistics over data set because ...
```

and exit with a negative return code (*i.e.*, `return -1;`).

Submit your work to http://marmoset01.shoshin.uwaterloo.ca project **L2-Statistics**. Your submission file must be named `Statistics.cpp`. Note that the filename is *case sensitive*.

Knowledge required to complete this exercise:

- Access to `argv[]` items
- Use of `atof()` functions
- Use of `while()` loop
- `float` variables and basic arithmetic operators
- Use of `cout`, `cerr`, `endl` and `std`

ECE 150 HOMEWORK 2: MORE STATISTICS AND INTEGRATION

SUBMISSION DUE WEDNESDAY, SEPTEMBER 19th AT 10:00 PM

Two Latin quotes in one document?

1. Running Statistics [1 marks]

The in-lab exercise required you to calculate the minimum, maximum, and average over a complete series of voltage readings. However, if what we are doing is continuous monitoring of the voltage on a line, there will never be an end to the voltage readings. In such a situation we need to keep *running statistics*: the on-going average, maximum, and minimum.

For this question, you are required to write a program that accepts several (one or more) command-line arguments that will be floating-point numbers representing a series of voltage readings. Valid voltage readings will be positive numbers less than 150. The list of readings will be terminated by a negative number. Your program should treat these as ongoing samples and for each such number it should compute the *current* minimum, maximum, and average. For example, if the program is invoked as

```
./RunningStats 110.3 112.9 108.1 109.2 -1
```

then the output should be:

```
Sample   Value  Minimum Average Maximum
1        110.3  110.3   110.3   110.3
2        112.9  110.3   111.6   112.9
3        108.1  108.1   110.433 112.9
4        109.2  108.1   110.125 112.9
```

You may assume that the arguments will be valid floating-point numbers within the range `[-FLT_MAX, FLT_MAX]` and you may use "`atof`" (ASCII-to-float, requires `<stdlib.h>`) to convert the arguments.

Submission: You should submit the result of this exercise to the project **H2-RunningStats**. Your submission file must be named `RunningStats.cpp`. Note that the filename is *case sensitive*.

2. Smoothing Statistics [1 marks]

There is a problem with the running-statistics method of the previous question: the statistics are based on the complete data history. This is a problem. For example, if we have a thousand voltage measurements and it was consistently between 110 and 115, with an average of 112, then in order for our program to register a voltage spike, say 135 V, we will need a large number of new samples.[1] A technique to address this is called the Exponentially Weighted Moving Average (EWMA). This technique works by applying the following formula to compute the average:

$$A_t = \alpha S_t + (1 - \alpha) A_{t-1}$$

where A_t is the average at "time" t, S_t is the sample taken at time t, A_{t-1} is the average computed for time $t - 1$, and $0 < \alpha < 1$ is a weighting parameter. When applying this technique, time is a discrete number and represents the successive samples (*i.e.*, time starts at 0 in increases by 1 for each new sample). To bootstrap this system we define:

$$A_0 = S_1$$

For this question, you are required to write a program that accepts several (one or more) command-line arguments. The first argument is the value of α; the remaining arguments are floating-point numbers representing repeated voltage-measurement samples, in the range of 0-150, terminated by a negative number. Your program should treat these as ongoing samples and for each such number it should compute the *current* exponentially weighted moving average per the desciption above. For example, if the program is invoked as

```
./RunningStats 0.125  110.3 112.9 108.1 109.2 115.2 -1
```

then the output should be:

```
Sample   Value  EWMA
1        110.3  110.3
2        112.9  110.625
3        108.1  110.309
4        109.2  110.171
5        115.2  110.799
```

You may assume that the arguments will be valid floating-point numbers within the range `[-FLT_MAX, FLT_MAX]` and you may use "`atof`" (ASCII-to-float, requires `<stdlib.h>`) to convert the arguments.

Errors and warnings should be dealt with appropriately, per the in-lab exercise.

Submission: You should submit the result of this exercise to the project **H2-EWMA**. Your submission file must be named `EWMA.cpp`. Note that the filename is *case sensitive*.

[1] Some questions to think about: how many voltage readings at 135 V would you need before the average voltage exceeded 125 V if you already have a sample size of 1000 with an average of 112 V? If readings are taken every 30 milliseconds, how long would the voltage be at 135 V before the average exceeded 125 V?.

3. Sliding-Window Statistics [1 marks]

While EWMA solves the problem of smoothly adjusting the average, aging out old readings, the maximum and minimum are still based on the total history. To solve this problem we can compute the maximum and minumum over a *window* of data and slide the window forward as new data arrives. In doing this, when the minimum or maximum age out, the new minimum or maximum is computed over a window, rather than simply being set to the current sample value. The sliding window size is expressed in terms of samples. For example, if we have a series of samples:

```
... 134.19 135.60 134.50 136.70 135.90 136.60 140.20 145.10 ... -1
    |<------------------------------>|
           |<------------------------------>|
                  |<------------------------------>|
```

then a sliding window of size 5 is as shown. The minimum, average, and maximum over each of these windows would then be:

```
Window Size: 5
Sample   Value   SWMinimum SWMaximum
...
...      135.90  134.19    136.70
...      136.60  134.50    136.70
...      140.20  134.50    140.20
...
```

For this question, you are required to write a program that accepts several (two or more) command-line arguments. The first command-line argument is the window size while the remaining ones will be floating-point numbers representing voltage readings and should be between 0 and 150, terminated by a negative number. Your program should compute the sliding window minimum, maximum of these numbers. The output of your program should be as shown above.

The program will be executed as follows:

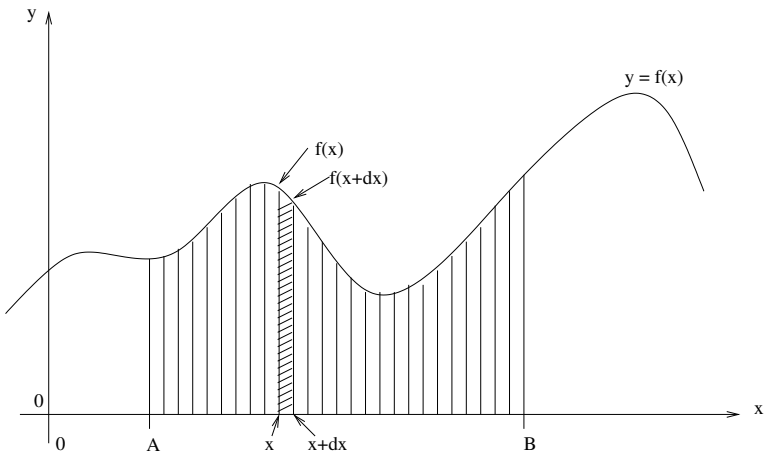
```
./SWStats <windowSize> <num1> <num2> ... -1
```

where "<windowSize>" is an integer in the range [INT_MIN,INT_MAX] and "<num1>", "<num2>", ... will be replaced with floating-point values within the range [-FLT_MAX, FLT_MAX] and you may use "atoi" and "atof" (ASCII-to-int and ASCII-to-float, requires <stdlib.h>) to convert the arguments.

Errors and warnings should be dealt with appropriately, per the in-lab exercise.

Submission: You should submit the result of this exercise to the project **H2-SWStats**. Your submission file must be named SWStats.cpp. Note that the filename is *case sensitive*.

4. Area Estimation [1 marks] Consider the problem of trying to determine the area under the function, $f(x)$ between $x = A$ and $x = B$, as illustrated below:



It may be the case that calculating the exact area is difficult or even impossible. In such a case, we can approximate it as follows. If we divide the area into a series of smaller areas, such as the one shown by cross-hatching above, between x and $x + dx$. We can approximate the area, A_x of that small section as:

$$A_x = dx * \frac{f(x + dx) + f(x)}{2}$$

The area can then be approximated by summing up all such areas between A and B.

$$\text{Area} = \sum_{x=A}^{x \leq B} A_x$$

For this problem, you are given a function declaration:

```
extern float f(float x);
```

that defines some function $f(x)$. You are required to write a program **Area.cpp** that takes as input two values, A and B, and computes the approximate area under the function $f(x)$ between A and B. The program is invoked as:

```
./Area <A> <B>
```

The output should be:

```
The area under f(x) from <A> to <B> is ...
```

You may assume $dx = 0.01$

You should handle errors and warnings appropriately.

Submit your work to http://marmoset01.shoshin.uwaterloo.ca project **H2-Area**. Your submission file must be named `Area.cpp`. Note that the filename is *case sensitive*.