Assignment SPIM 1

Due Date: February 22

Purpose
You will use SPIM, our MIPS emulator, to write some code in assembly language. In this project, you will use arrays, loops, and conditionals. It’s just like Comp 115, only much more fun!

Problem
Given a list of up to 25 positive integers, we wish to find some basic statistics and display the results in a nice way.

Input
The user will input a list of up to 25 positive integers in the range of \{1..20\}, ending with a negative value. Note that the same number may appear multiple times. Prompt the user only once.

Output
The program should display:

- the number of values input
- the integer mean
- the variance, $s^2$. This is calculated as follows:

\[
s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2
\]

where $n$ = the number of values, $x_i$ is the $i^{th}$ value in the list, and $\bar{x}$ = mean of all the values.

- a bar graph that shows the frequency of each value. For example, if the range were \{1..4\} and the 1 was seen twice, 2 was seen once, 3 was not seen, and 4 was seen three times, then a suitable bar graph, where a 0 is a placeholder symbol, would look like:

\[
\begin{align*}
1 & \quad 00 \\
2 & \quad 0 \\
3 & \\
4 & \quad 000
\end{align*}
\]

- display a message, such as “Program completed” after the graph indicating that your program has terminated properly.

Be sure to label all output; display all of the output values as integers. You may display any suitable placeholder symbol in your graph.
Specifics

- Since we have not yet covered floating point operations, all computations should be done using integer arithmetic.

- You will need at least one array in your solution. To do this, create one or more arrays of at least 25 elements in the data portion of the program, using arbitrary integers. This will allocate storage for any data to come.

- You must have a good introductory comment including your name(s), the program file name, a description of the program, a description of the input, and a description of the output. Comment registers as well as possible, so that a reader can figure out what each register holds. Of course, with a limited number of registers, some may be reused, so comment the best you can. Although we can’t write true functions yet, you can group code together and use jumps to simulate functions. Comment each of these groups in a general way (e.g., “Find mean of the list.”). Finally, line up and use good spacing in the assembly code in some consistent way, so that it is as readable as possible.

Notes

For full credit, turn in your source code via email before 11:59:59 PM of the 22nd. Name your file lastNameSPIM1.a; for example, my submission would be called gousieSPIM1.a. You may turn in more than one version, but I will grade only the latest submission.

Turn in a printed copy in class on February 23. Write/type and sign the Honor Code on the hard copy: “I have abided by the Wheaton College Honor Code in this work.”

This is a fairly trivial problem in Python or C++, and I encourage you to first write the solution in one of those languages. You can then translate that program to SPIM. Nevertheless, it will probably take longer than you expect to get it working in assembly language. Do things in small stages, making sure everything works before moving on to the next stage.

Computers in the future may weigh no more than 1.5 tons.

– Popular Science magazine, 1949