Assignment SPIM 1

Due Date: March 5

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 like a time-machine going back to Comp 115!

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 maximum value
- the integer mean
- the integer remainder found when computing the mean
- the mode (the value(s) seen most often)
- a bar graph that shows the frequency of each value. For example, if the range were {1..4} and 1 was seen twice, 2 was seen once, 3 was not seen, and 4 was seen three times, then a suitable bar graph, where + is a placeholder symbol, would look like:

  ```
  1 ++
  2 +
  3
  4 +++
  ```

- display a message, such as "Program completed" after the graph indicating that your program has terminated properly (i.e., the program is not in an infinite loop).

Be sure to label all output; display all of the output values as integers. You may display any suitable placeholder symbol in your graph. The graph should be properly aligned.

Specifics

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

- To find the mode, you will need one or more arrays. To do this, create 25-element arrays in the data portion of the program, using arbitrary integers. This will allocate storage for any data to come.
- You must include a good introductory comment including:
  - your name
  - a description of the program
  - a *complete* description of the input
  - a *complete* description of the output
- For your own sanity, 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 the mean of the list."). Finally, line up the assembly code in some consistent way, so that it is as readable as possible.

Notes

This is a fairly trivial problem in Python or C++, but it will take longer than you expect to get it working in assembly. I encourage you to write the solution in one of those languages first, and then translate your program to SPIM. In SPIM, write a (very) small section of code at a time and sure everything works before moving on.

Add the Honor Code in a comment near the top that includes your electronic "signature." Turn in your source code via email as an attachment by 11:59:59 PM on the due date. Name your file with your first initial followed by your last name and finally "SPIM1.a" or "SPIM1.s" as in `mgousieSPIM1.a`. Note there are **no spaces** in the filename.

Computers are good at following instructions, but not at reading your mind.
– Donald Knuth