Assignment MIPS 1

Due Date: February 23

Purpose

You will use MARS, 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 - except it's now in what looks like gibberish!

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 zero (the sentinel value). Note that the same number may appear multiple times. Prompt the user only once. You may assume that all data is input correctly; that is, you do not have to do error checking.

Output

The program should display:

- the number of values input (not including the sentinel value)
- the maximum value
- the integer mean
- the modulo (integer remainder found when computing the mean)
- the mode (the value seen most often). You may assume that the mode of the data set is unique.
- the sum of all the values ≤ 10
- the sum of all the values > 10
- 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 +++
```

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 labeled and **aligned**; for example, bars representing equal frequencies should always be evenly aligned.

Specifics

- Since we have not yet covered floating point operations, all computations should be done using integer arithmetic.
- You must include a good introductory comment including:
  - your name
  - the filename
  - a general description of the program
  - a *complete* (specific) description of the input
  - a *complete* (specific) 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, and use blank lines/comments to delineate code sections, so that the program 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 into MIPS. In MIPS, write a (very) small section of code at a time and be sure everything works before moving on.

The name of your source code file should be your first initial + last name + project number **with no spaces**, as in `mgousieMIPS1.s|asm`. Submit your source code via Canvas by 11:59:59 PM on the due date.

Turn in a printed copy of your code in class on February 24^{th}. Write/print and **sign** the Wheaton Honor Code Pledge on what you turn in: "I have abided by the Wheaton College Honor Code in this work."

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