Purpose: Conditionals and loops in MIPS.

Details: You will submit two MIPS programs. The programs must be fully commented; comments should include your name, the lab number, the problem number, the lab date, the Honor Code pledge, and a description of what the program does. The MIPS code must be documented for each part of the program. Imitate the commenting style of the sample programs.

1. [Modify a program.] In the class shared repository, in folder lab5, get a copy of the program “slowmult.asm.” Read the comments to learn what it does, make sure you can run it successfully.

   Now modify the program so that it uses the smaller of the two input values as the “counter.” For instance, if the inputs are 50 and 10, the loop should perform 10 additions of the value 50 rather than 50 additions of the value 10. The easiest way to do this is to use comparison instructions in MIPS (slt, beq, bne) and the unconditional jump (j) to swap the values of the user’s two inputs if the second is smaller than the first. (Obviously if the two values are equal then either one may be used, so don’t do the swap in this case.) Use my code for multiplying—all of your changes will be made in the code preceding the loop. Don’t write a second loop! Just swap the values, if necessary, so that the first input is less than or equal to the second.

2. [Write your own program.] Write a MIPS program that does the following:
   - Prompts the user to enter a positive integer n less than or equal to 20
   - Prompts the user to enter n more integers, one per line
   - At the end of the input process, print the maximum and the minimum value among the n inputs, appropriately labeled. Your output should imitate the following style:
   
   Enter n between 1 and 20: 4
   Enter 4 integers, one per line:
   -5
   10
   -17
   8
   The minimum is -17, the maximum is 10

   In particular, the value of n entered by the user should appear in the prompt for the values.
You do not have to store the values in memory—you can discard them once they have been read in and compared to the current max and min.

This is not a hard problem, but it is highly prone to error! Take your time, don’t try to get too fancy, step through it. And most important, *ask questions if you are stuck!*

3. Push your programs, fully commented as described above, into the repository you shared with me.

*Questions about the lab? Bring them to class on Tuesday morning!*