[redacted] Computer Architecture and Assembly Language
Programming Assignment #3

Objectives: more practice with

1. Implementing data validation
2. Implementing an accumulator
3. Integer arithmetic
4. Defining variables (integer and string)
5. Using library procedures for I/O
6. Implementing control structures (decision, loop, procedure)

Description:

Write and test a MASM program to perform the following tasks:

1. Display the program title and programmer's name.
2. Get the user's name, and greet the user.
3. Display instructions for the user.
4. Repeatedly prompt the user to enter a number. Validate the user input to be in [-100, -1] (inclusive). Count and accumulate the valid user numbers until a non-negative number is entered. (The non-negative number is discarded.)
5. Calculate the (rounded integer) average of the negative numbers.
6. Display:
   i. the number of negative numbers entered (Note: if no negative numbers were entered, display a special message and skip to *iv.*)
   ii. the sum of negative numbers entered
   iii. the average, rounded to the nearest integer (e.g. -20.5 rounds to -20)
   iv. a parting message (with the user's name)

Requirements:

1. The *main* procedure must be modularized into commented logical sections (procedures are not required this time)
2. The program must be fully documented. This includes a complete header block for identification, description, etc., and a comment outline to explain each section of code.
3. The lower limit should be defined as a constant.
4. The usual requirements regarding documentation, readability, user-friendliness, etc., apply.
5. Submit your text code file (*.asm*) to Canvas by the due date.

Notes:

1. There are no new concepts in this programming assignment. It is given for extra practice, to keep MASM fresh in your mind while we study internal/external data representation.
2. This is an integer program. Even though it would make more sense to use floating-point computations, you are **required** to do this one with integers.

Example (see next page)

Example (user input in ***italics***):

```
Welcome to the Integer Accumulator by Austin Miller
What is your name? Caleb
Hello, Caleb

Please enter numbers in [-100, -1].
Enter a non-negative number when you are finished to see results.
Enter number: -15
Enter number: -100
Enter number: -36
Enter number: -10
Enter number: 0
You entered 4 valid numbers.
The sum of your valid numbers is -161
The rounded average is -40
Thank you for playing Integer Accumulator! It's been a pleasure to meet you, Caleb.
```

Extra-credit options (original definition must be fulfilled):

1. Number the lines during user input.
2. Calculate and display the average as a floating-point number, rounded to the nearest .001.
3. Do something astoundingly creative.

To ensure you receive credit for any extra credit options you did, you must add one print statement to your program output PER EXTRA CREDIT which describes the extra credit you chose to work on. You will not receive extra credit points unless you do this. The statement must be formatted as follows...

```
--Program Intro--
**EC: DESCRIPTION

--Program prompts, etc—
```