

Course 2 Unit 3 Programming Assignment 3: When Will They Stop?

Assignment Description

You want to develop a program that takes in as many integers as the user wants to enter and then tells them how many integers they entered and the mean (average) of those numbers. This will also give you more practice implementing while loops!

In this assignment, you'll get integer numbers until you get a -1, at which point you'll print how many numbers you processed (not including the -1) and the mean of those numbers (not including the -1).

Getting Started – Clone Your Repository

1. Click the appropriate link below and accept the assignment to create your copy of the repository with the starter code and submitting your work:
 - a. Gallant AM: <https://classroom.github.com/a/kauOoNuO>
 - b. Gallant PM: <https://classroom.github.com/a/EIPksxN>
 - c. Nunn AM: <https://classroom.github.com/a/YuY2T6il>
 - d. Nunn PM: <https://classroom.github.com/a/ijbuTJ9Q>
 - e. Wijaya AM: <https://classroom.github.com/a/HQ8LAvlg>
 - f. Wijaya PM: <https://classroom.github.com/a/5iRNUxZ2>
2. In GitHub Desktop, clone the repository to your desktop.
3. Open the project in Visual Studio.
 - a. **Important:** You MUST only add code as indicated by the comments in that file. If you don't, you're virtually guaranteed to fail all the test cases in the automated grader.

Requirements

The code I've provided in the **Program.cs** file reads in a set of integers that ends with -1. Your code must call the **GetValue** method and store the **int** the method returns in a variable, then process that number as appropriate.

Your solution must print the following on a single line:

- The count of the numbers you processed (not including the -1)
- A space
- The mean of the numbers you processed (not including the -1)

For example, if the set of numbers you process is 1 2 3 4 -1, your output should be the following on a single line:

```
4 2.5
```

We'd typically label our output to tell the user what the output means, but that will just confuse the automated grader. You must print **ONLY** the values described above in your output, with the values appearing on a single line with a single space between them.

Note: For the special case when -1 is the first number, your count and mean should both be set to 0.

Running Your Code

Because of the code I included to work with the automated grader, when you run your program the command prompt window will open and it will sit there doing nothing. To make your code run, type in a set of integers, ending with -1, with a space between each integer, and press the **<Enter>** key; your code should then run so you can check your output. For example, your input could be:

```
1 2 3 4 -1
```

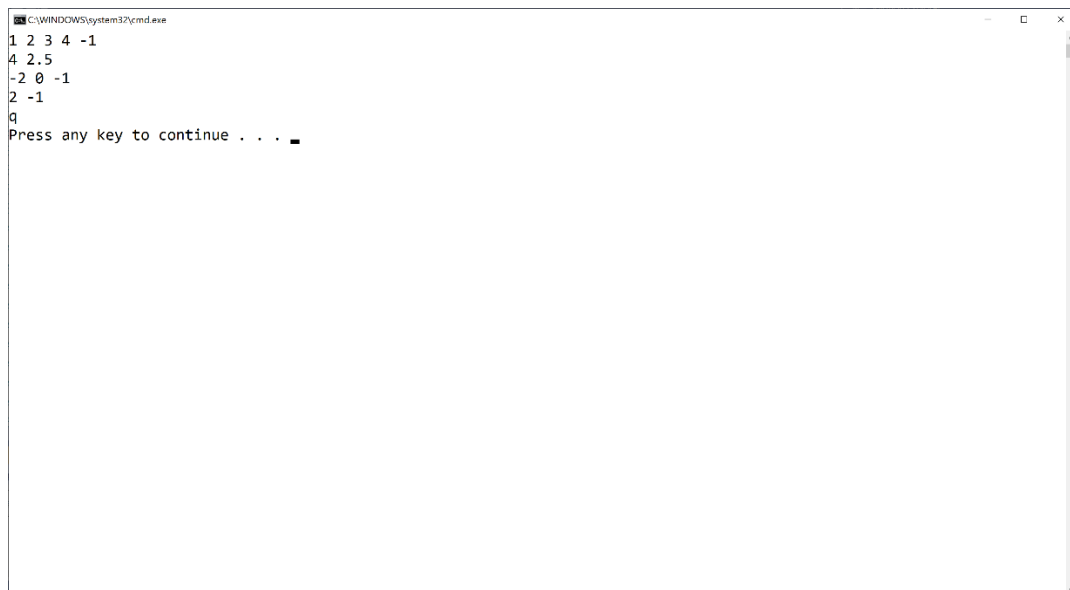
You can actually run your code again if you want to by typing in a set of integers, ending with -1, with a space between each integer, and pressing the <Enter> key again. When you're ready to stop running your code, type q (for quit).

Here's what running the code multiple times with different inputs should look like (remember, you're including a newline at the end of each of your output lines). The first line is the integer input, the second line is your output line, and so on:

```
1 2 3 4 -1
4 2.5
-2 0 -1
2 -1
q
```

Important Note: The CodeHS formatting makes it look like there's a blank line between each of the lines above, but there's not. The above output should be exactly 5 lines of output.

The image below shows my console window when I run the code multiple times as described above:



```
C:\WINDOWS\system32\cmd.exe
1 2 3 4 -1
4 2.5
-2 0 -1
2 -1
q
Press any key to continue . . .
```

If your output doesn't match the image above EXACTLY (no extra words, characters, spaces, or blank lines) you'll fail all the test cases in the automated grader.

Test Case Inputs

The automated grader uses the following set of test case inputs to test your code (1 test case per line):

```
1 2 3 4 -1
-1
-2 0 -1
1 1 1 1 1 -1
2 1 0 1 2 -1
-1
-2 -3 -4 -1
-2 0 2 -1
5 5 5 5 5 5 5 5 5 1 -1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 -1
```

You should figure out what the expected results are for each test case input and make sure your code is generating those expected results in the required format before submitting your code for grading.

Common Problems

Here are a couple things you should check to make sure you are getting the correct results:

1. You have to start with the Visual Studio project I gave you in the assignment materials and add your code in the section indicated. The code I gave you includes the appropriate structure for the automated grader to work. If you don't do this, you'll almost definitely fail all the test cases in the automated grader.
2. I give you an example in this assignment (and all assignments) for what output you should get when you run the code multiple times. Be sure to try that example; if your code doesn't generate output exactly as shown in the example, your code isn't working properly.

Submit Your Work

1. Make a final test of your code and copy the output from the terminal window.
2. If you needed to make any additional changes to your code, make sure you commit them.
 - a. By committing and pushing your updates to GitHub you have submitted your assignment on GitHub Classroom.
3. Return to CodeHS. Paste your output into the code window to complete the assignment.