Assessment 1 Report

IMPLEMENTATION

For this section I will explain my code from top to bottom. Initially, I initialised variables containing the strings that the user will be prompted with. I also initialised variables called:

- `loops` – The number of loops to complete; defined when the user when prompted by: “How many numbers”.
- `number` – The number entered by the user when prompted by: “Enter a number”.
- `positives` – The counter for positive numbers.
- `negatives` – The counter for negative numbers.
- `zeroes` – The counter for zeroes.

The flow of execution is as follows:

1. First the program will print the contents of the string `msgMany` to the console and store the user’s input to `loops`.
2. `loops` will then be moved to ECX before starting the `inputLoop` label.
3. To avoid ECX getting poisoned by external functions ECX is pushed to the stack.
4. `msgEnter` will then be printed to the console and the user’s response saved to `number`.
5. `number` will then be compared to 0:
    a. If the comparison sets the zero flag, then execution will branch to `addZero`.
        i. `addZero` will load the `zeroes` variable into the accumulator, increment it, and then move the accumulator back to the `zeroes` variable.
    b. If the comparison set the sign flag, then execution will branch to `addNegative`.
        i. `addNegative` will load the `negatives` variable into the accumulator, increment it, and then move the accumulator back to the `negatives` variable.
    c. If none of those flags are set, then the execution falls through and the `positives` counter is incremented.
6. All branches jump to the `back` label.
7. The loop counter is popped from the stack and loops to step 3., or falls through, using `loop.`
8. The summary is printed by:
    a. Each message and value to be displayed is pushed to the stack interlaced with a loop counter.
        i. The loop counter is pre-set in the stack so that the loop counter doesn’t get in the way when calling `printf` later.
    b. `msgBar` is printed and the stack pointer incremented, by 1 word, to the next item.
    c. A loop is started to print all the messages and values from the stack. After the item is printed the stack pointer is incremented 2 words to reveal the loop counter.
        i. The loop counter is popped into ECX. Program flow loops to step 8c., or falls through, using `loop.`

TESTING

Single Positive Input

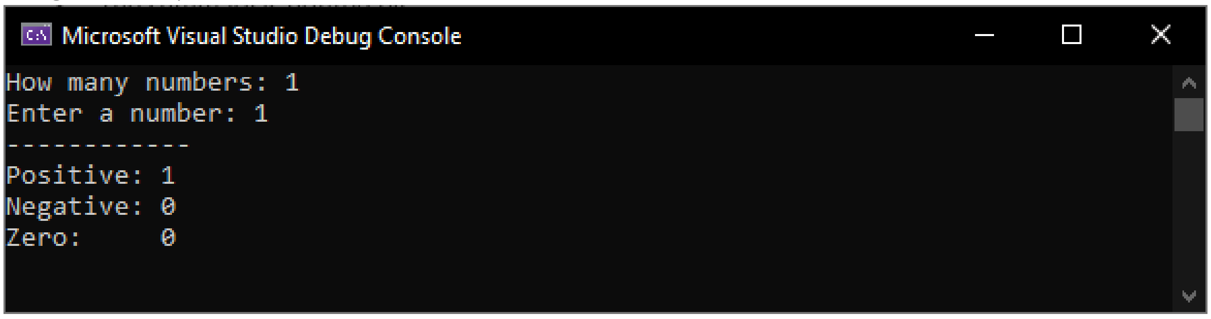
Input

Iterations	1
Number(s)	1

Expected Output

Positive	1
Negative	0
Zero	0

Program Output



Microsoft Visual Studio Debug Console

```
How many numbers: 1
Enter a number: 1
------------
Positive: 1
Negative: 0
Zero:     0
```

Long Multi-Input

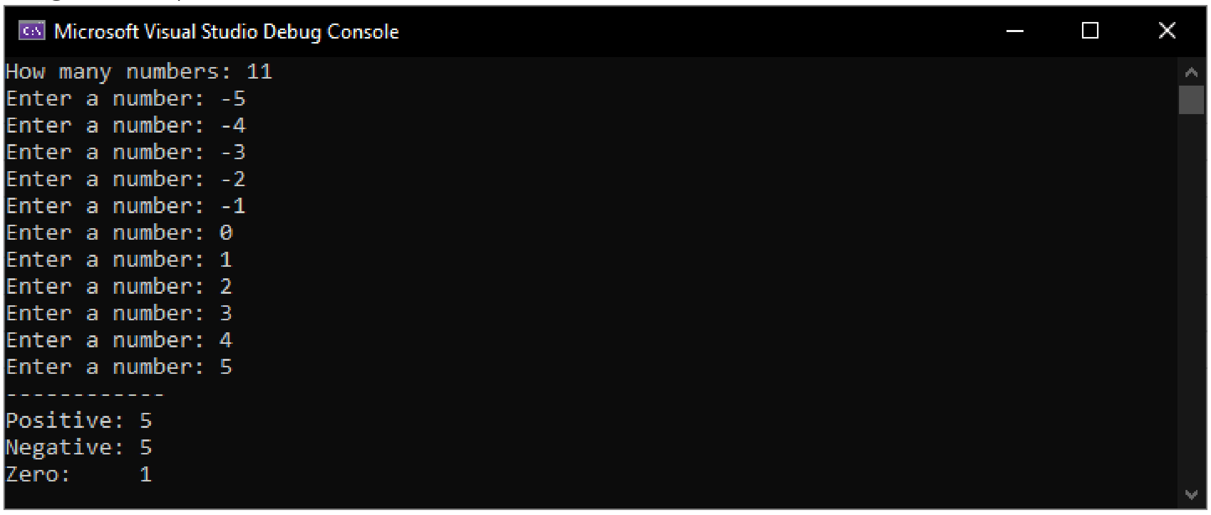
Input

Iterations	11
Number(s)	-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5

Expected Output

Positive	5
Negative	5
Zero	1

Program Output



Microsoft Visual Studio Debug Console

```
How many numbers: 11
Enter a number: -5
Enter a number: -4
Enter a number: -3
Enter a number: -2
Enter a number: -1
Enter a number: 0
Enter a number: 1
Enter a number: 2
Enter a number: 3
Enter a number: 4
Enter a number: 5
------------
Positive: 5
Negative: 5
Zero:     1
```