# ASSIGNMENT-1
# MAINSTREAM

[ASM'19 CSW-353]

Assignment: 1 (Mainstream)
Due date: 8 November 2019 Midnight
Course: Assembly language CSW-353


Faculty of Computer and Information Sciences
Instructors: Dr.Karim Emara
Dr.Salsabil Amin

---

## General instructions:

Regarding your file:

1) Submit **only running** code that you have tested before.
2) Use the provided template to write your code. Do not use any other templates.
3) Submit **only** the content of **main.asm** file. You have to press "Submit" button in the form.
4) Submit your code through this Google form link
5) Do **not** use: Conditional directives, such as .if, .while, etc …
6) Do **not modify** test cases.
7) **Remove** "call panic" from any implemented procedure.

**This assignment is intended for individual contribution. Sharing ideas or part of answers is considered plagiarism and will not be tolerated. All submissions will be checked for plagiarism automatically.**

Marks will be **deducted** for not following any of the above submission rules (-5/instruction).

**PLEASE READ THE AUTOGRADER NOTES CAREFULLY BEFORE SUBMITTING YOUR CODE**

---

Notes:

1. All the sample runs were taken from the assignment's template.
2. Array values are not read in one line. They are read line-by-line.

---

*Good luck*
*Assembly team 2019*

Assignment: 1 (Mainstream)
Due date: 8 November 2019 Midnight
Course: Assembly language CSW-353


Faculty of Computer and Information Sciences
Instructors: Dr.Karim Emara
Dr.Salsabil Amin

---

**Q1.** In the procedure called "Q1", write assembly code that reads a number that represents the number of rows for a triangle, then accordingly draw the following triangle structure:

Sample input:
5

Output:
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15

**Sample Run:**

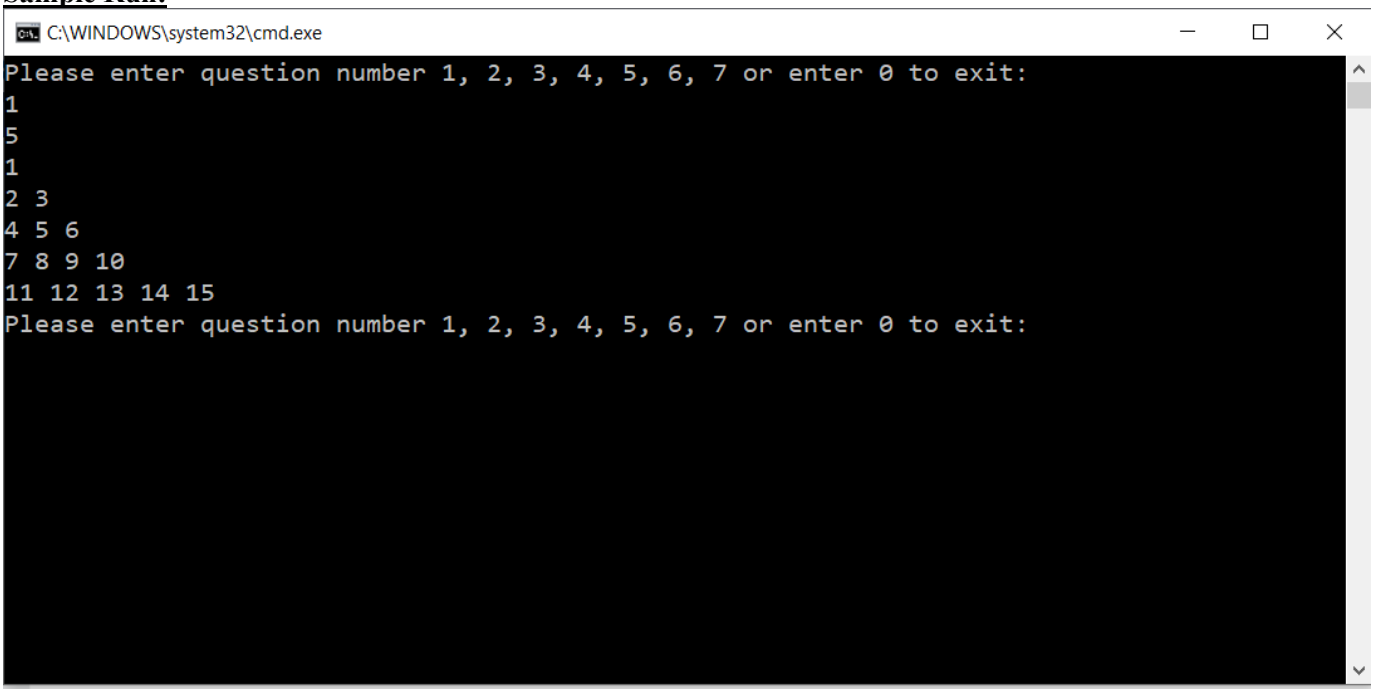


*Figure 1 question-1 sample run*

*Good luck*
*Assembly team 2019*

**Q2.** In the procedure called "Q2", write an assembly program that calculates the first user input values of the Fibonacci number sequence.

**Constraints:**
Maximum number of user input is 20.
You do not have to check if it's larger than 20.

**Sample Input:**
20

**Sample Output:**
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765

**Sample Run**

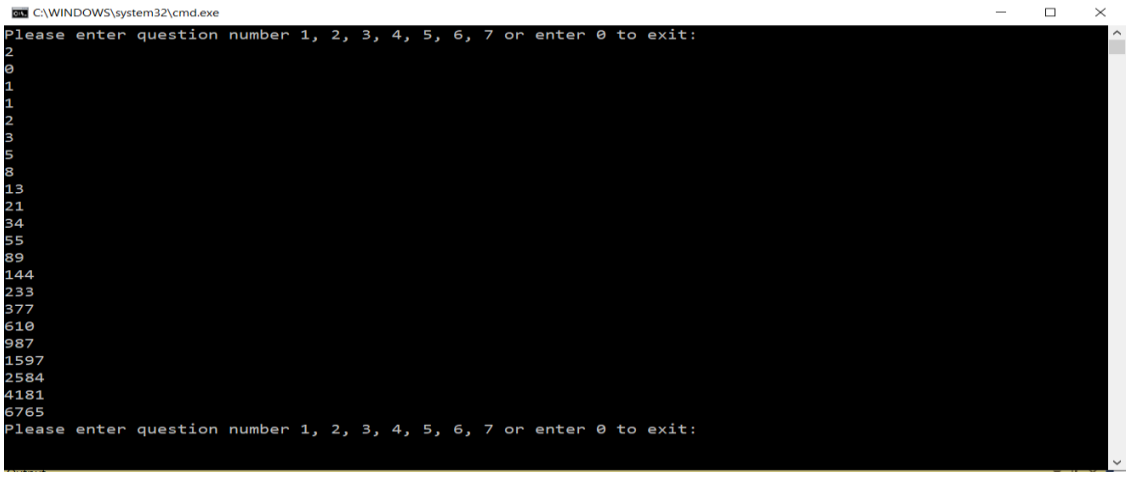


*Figure 2 question-2 sample run*

**Q3.** In the procedure called "Q3", write an assembly code to add individual array items N times. The array length is K where 1 <= K <= 5 and should be input from the user. The number of **possible reading cases is 2**.

**Constraints:**

- The first line of input contains an integer N, denoting the number of adding times (Should be positive).
- The second line of input contains an integer K, denoting the length of the array (Max 5 elements).
- The third input is the array items, one by one.

**Sample Run:**

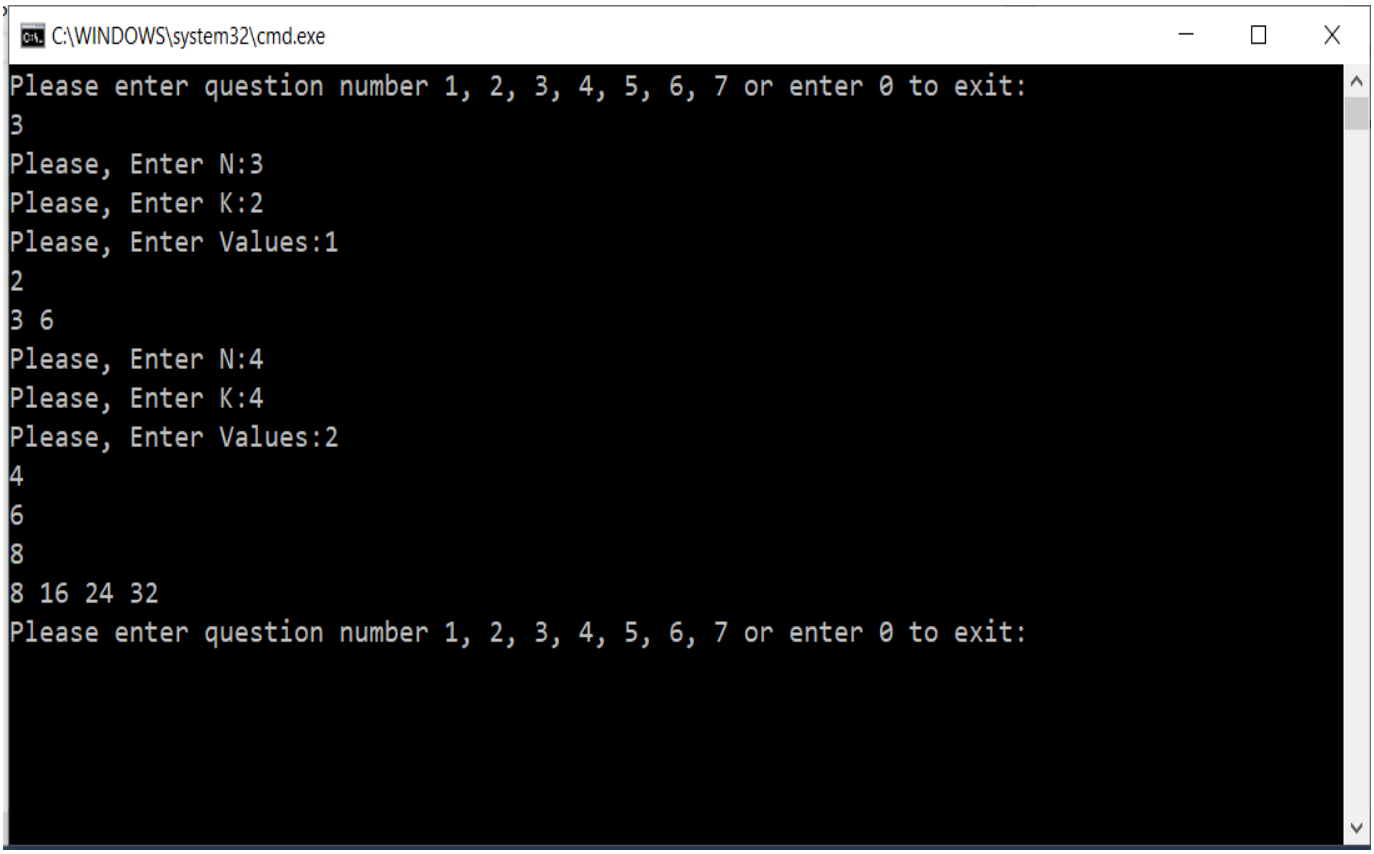


*Figure 3 question-3 sample run*

*Good luck*
*Assembly team 2019*

**Q4 .** In the procedure called "Q4", write assembly code to reverse each row in a fixed triangle of numbers. The triangle of numbers will be read from user and stored in an array named Q4_Matrix_1 and the procedure should reverse its rows and store it in Q4_Matrix_2.

**Constraints:**

- **Your matrices data type should be byte**

**Sample Run:**

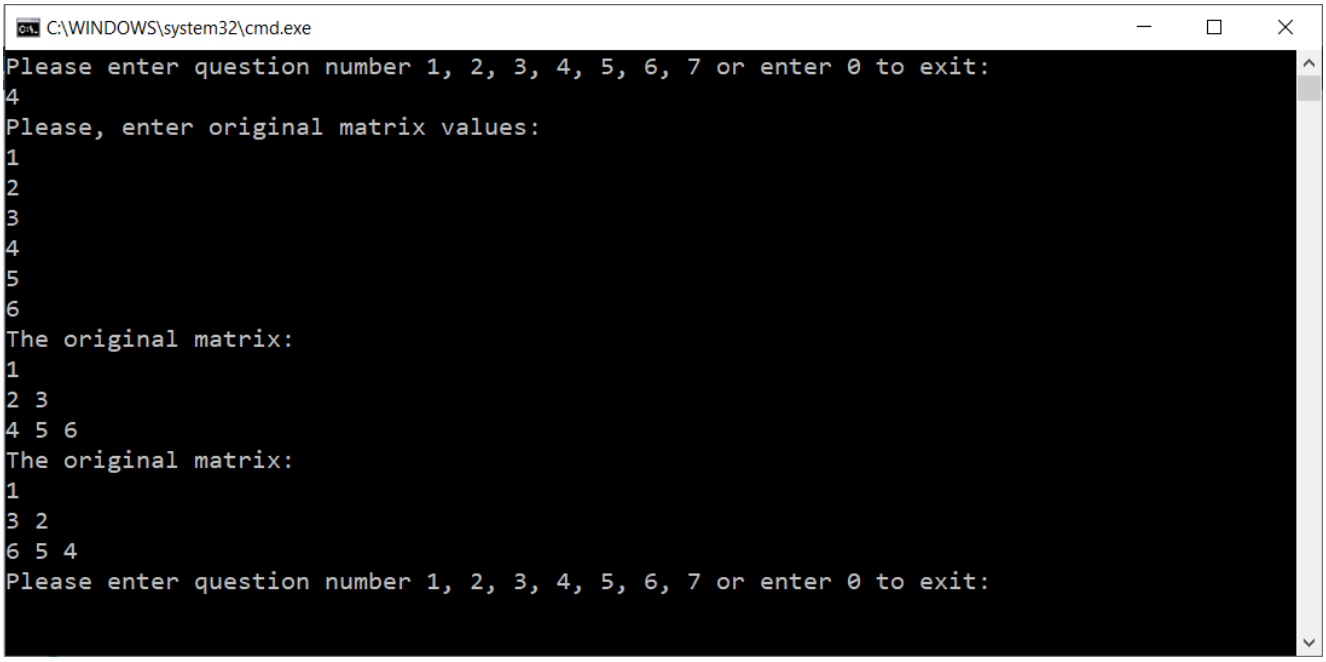


*Figure 4 quesiton-4 sample run*

*Good luck*
*Assembly team 2019*

**Q5.** In the procedure called "Q5", write an assembly code that calculates the number of dollars you have to borrow. If you want to pay **w items**, you need to pay **k dollars** for the first item, 2k dollars for the second one and so on (in other words, you have to pay i*k dollars for the i-th item). You have **n dollars**. How many dollars do you have to borrow from your friend to buy w items?

**Sample Run:**

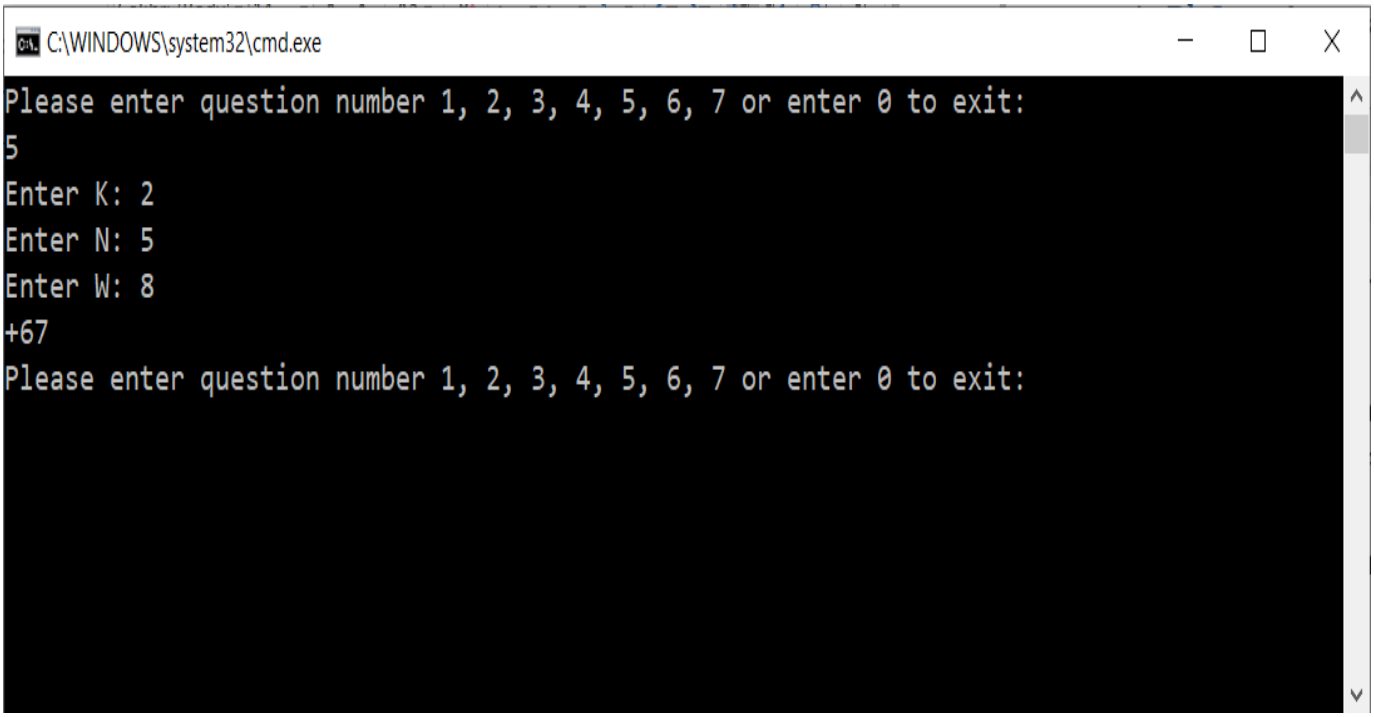


*Figure 5 question-5 sample run*

*Good luck*
*Assembly team 2019*

**Q6.** In the procedure called "Q6", write an assembly code that generates n lucas number. Lucas numbers are defined as the sum of its two immediately previous terms. But the first two terms are constant 2 and 1. The user should enter n which is the number of lucas numbers to be displayed.

**Constraints:**

- Minimum number of elements to be displayed is 5.
- The Lucas numbers are in the following integer sequence:

  2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123 ……..

**Sample Run:**

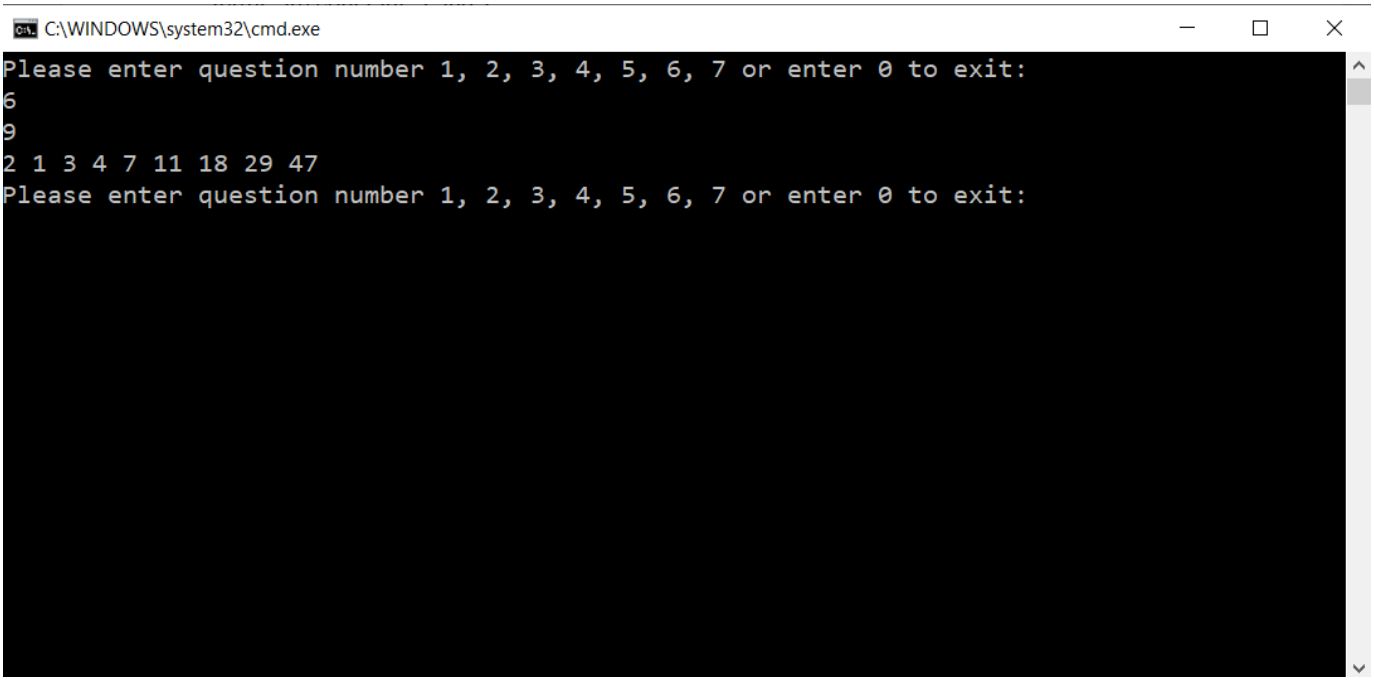


*Figure 6 question-6 sample run*

*Good luck*
*Assembly team 2019*

**Q7.** In the procedure called "Q7", write assembly code that calculates the following operations on an array of DWORD integers. First, multiply each integer in the array by its 1-based index and then summing them all together. Read the array size from the user, then read the numbers into an array. The maximum number of elements in the array is 20. Do not use the MUL instruction.

For example, in the sample run, the result is calculated as follows:
(2 * 1) + (50 * 2) + (175 * 3) = 627,
Where we multiplied each number in the array (2, 50, 175) with its position respectively (starting with 1).

**Constraints:**

- All input numbers are integers.
- Starting index should equal 1.

**Sample input:**

3 for the array size
2
50
175

**Output:**

627

**Sample Run:**

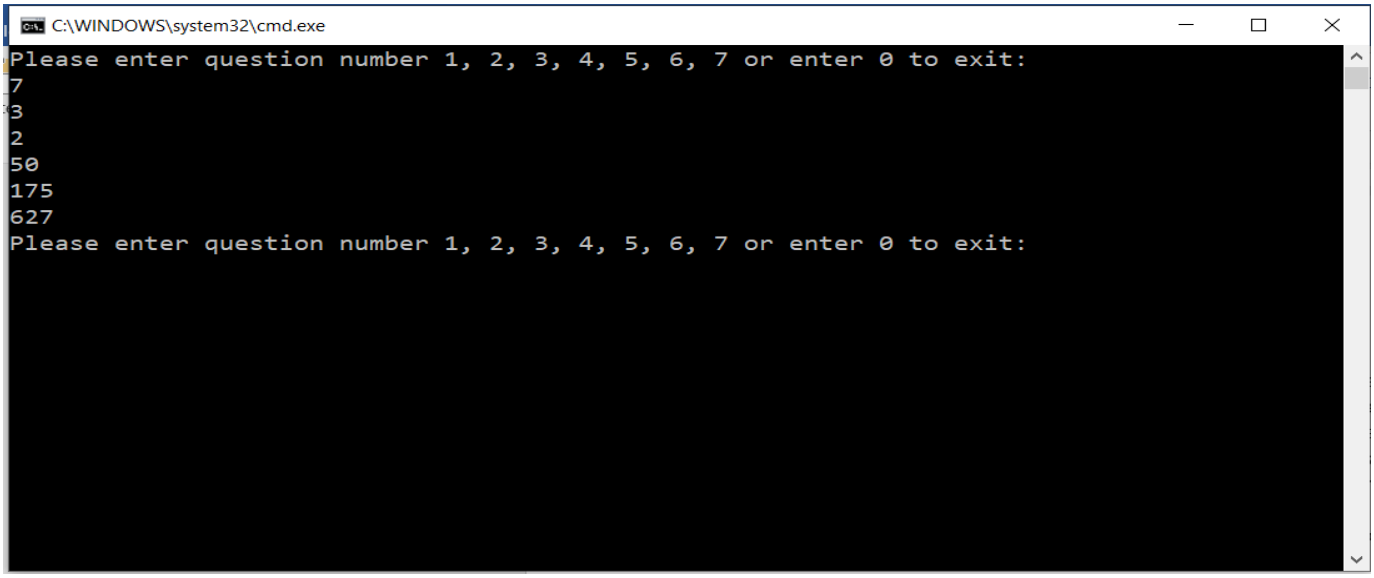


*Figure 7 question-7 sample run*

*Good luck*
*Assembly team 2019*

## Hints:

- `WriteInt` is an Irvine function that prints an integer value that must be stored in EAX register (number's sign is printed).
- `ReadInt` is an Irvine function that reads an integer from the keyboard and stores it in EAX register (the input integer is signed).
- `WriteDec` is an Irvine function that prints an integer value that must be stored in EAX register (number's sign is not printed).
- `ReadDec` is an Irvine function that reads an integer from the keyboard and stores it in EAX register (the input integer is not signed).
- `WriteChar` is an Irvine function that prints a character that must be stored in AL register.
- `ReadString` is an Irvine function that reads a string from the keyboard, stopping when the user presses the Enter key. Pass the offset of a buffer in EDX and set ECX to the maximum number of characters the user can enter. The procedure returns the count of the number of characters typed by the user in EAX.
- `WriteString` is an Irvine function that writes a string to the console. Pass the offset of a buffer in EDX.
- `ReadHex` used to read a hexadecimal value from the user. The value after the read is stored in EAX register.
- `WriteHex` used to write a hexadecimal value to the screen. The value to be displayed is stored in EAX register before calling this procedure.
- `ReadChar` is used to read a char from the console. The value read from the console is placed in "al" register.
- More about these functions and similar ones can be found in section 5.3 of the book.

*Good luck*
*Assembly team 2019*