Q1. In the procedure called "Q1", write an assembly code that swaps two characters in a word. The program should read 2 indices (1-based) of the two characters to swap and the word. The program should display the word again with the characters swapped.

Constraints:

- The string will not exceed 30 Character.

Sample Input:

4

2

Hello

1

14

Assembly Is Awesome

Sample Output:

Hlleo

wssembly Is AAesome

Sample Run:

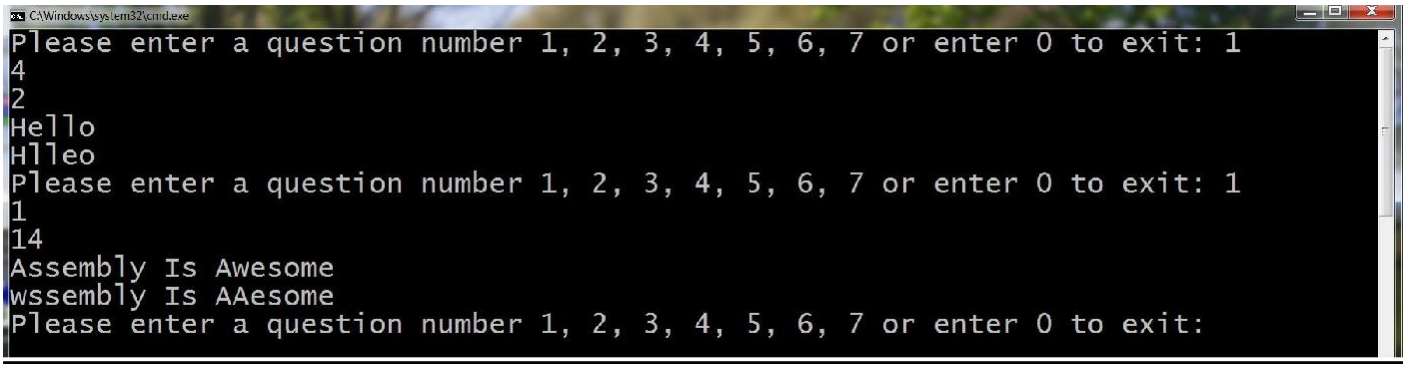

Figure 1 question-1 sample run

Q2. In the procedure called "Q2", write assembly code that reads an **even** number **N** and a count **M.** You should print the **M** even and the **M** odd numbers following the given number **N.** (Hint: use ReadDec, WriteDec and WriteChar procedures. Refer to the hints section).

Constraints:

- **N** is always even.
- $1 \leq N \leq 100, 1 \leq M \leq 100.$
- Print the **even** sequence first then the **odd** sequence.

Sample Input:

4

6

8

5

20

4

Sample Output:

6 8 10 12 14 16
5 7 9 11 13 15

10 12 14 16 18
9 11 13 15 17

22 24 26 28
21 23 25 27

Sample Run

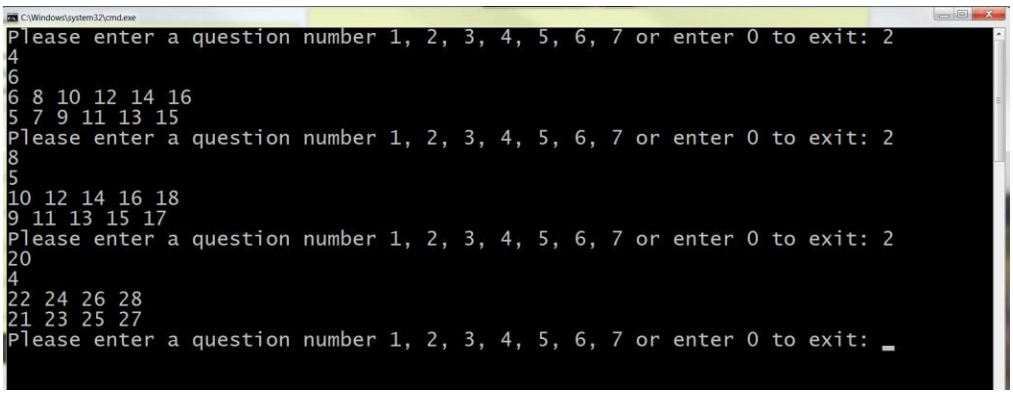

Figure 2 question-2 sample run

Q3. In the procedure called "Q3", write an assembly code that takes an input number and calculates its power of two.

Constraints:

- **DO NOT** use **MUL** Instruction.
- Numbers are only **positive**.
- Zero and one are not included.
- Maximum input number is **10,000.**

Sample Input:

2

4

8

16

Sample Output:

4

16

64

256

Sample Run

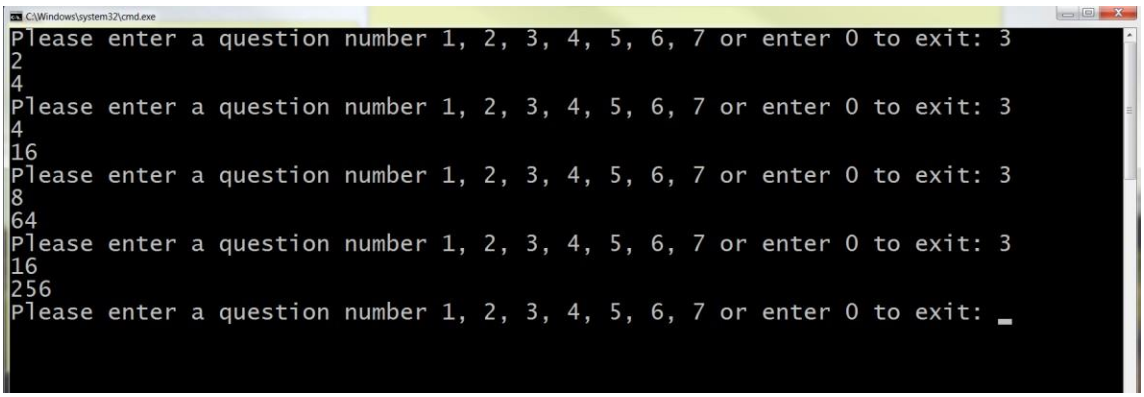

Figure 3 question-3 sample run

Q4 . In the procedure called "Q4", write assembly code that reads a string and an integer **n** and returns a string made of the first and last **n** characters from the string.

Constraints:

- Maximum input string length is 50 characters.
- $1 \leq n \leq 20$.

Sample Input:

Welcome
3

Assembly
2
Hello World
4

Sample Output:

Welome

Asly

Hellorld

Sample Run:

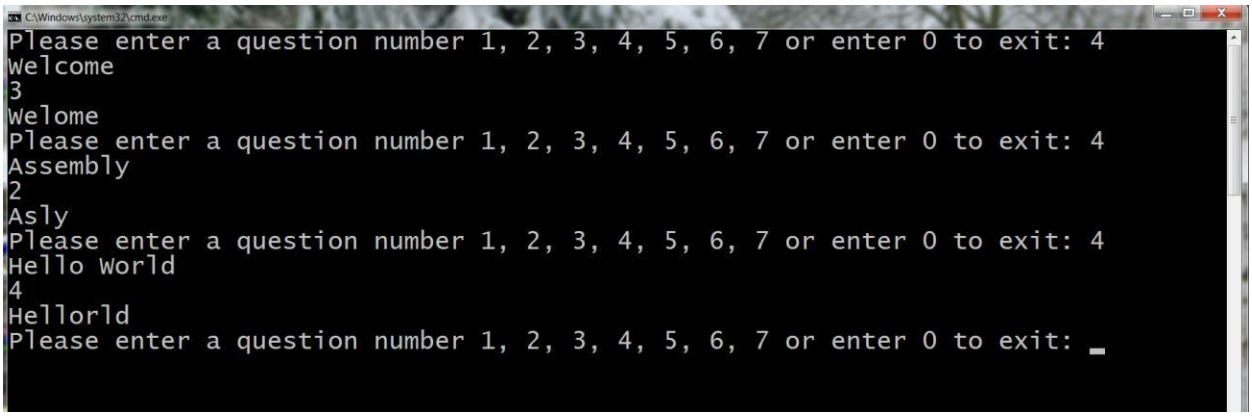

Figure 4 quesiton-4 sample run

Q5. In the procedure called “Q5”, write an assembly code that inserts a string into another string. The program should read 2 strings and the starting index (0-based) to insert in.

Constraints:

- The input strings will not exceed 30 characters.

Sample Input:

WelcomeAssembly

To

7

HelloYear

Third

5

Sample Output:

WelcomeToAssembly

HelloThirdYear

Sample Run:

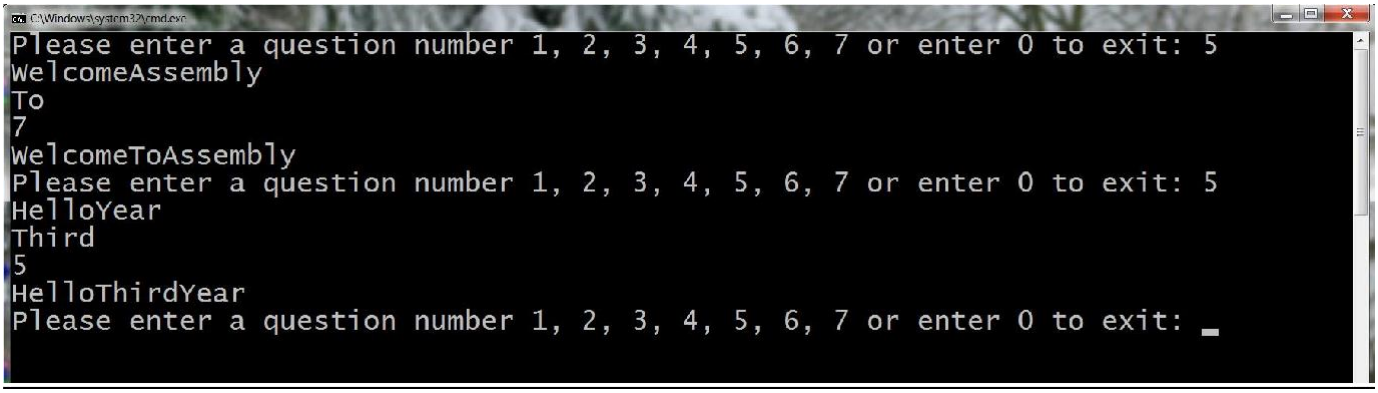

Figure 5 question-5 sample run

Q6. In the procedure called "Q6", write a function that combines two lists (same length) by alternatingly taking elements. Your program should read the size of the list first. Next, read both lists by reading character by character from the console (use ReadChar procedure refer to hints section).

Constraints:

- Maximum list size is 50 elements.
- .Data
  LengthofList dword ?
  List1 byte 50 dup (?)
  List2 byte 50 dup (?)
  OutputList byte 100 dup (?)

Sample Input:

3

1

2

3

A

B

C

5

A

B

C

D

E

1

2

3

4

5

Sample Output:

[1,A,2,B,3,C]

[A,1,B,2,C,3,D,4,E,5]

Sample Run:

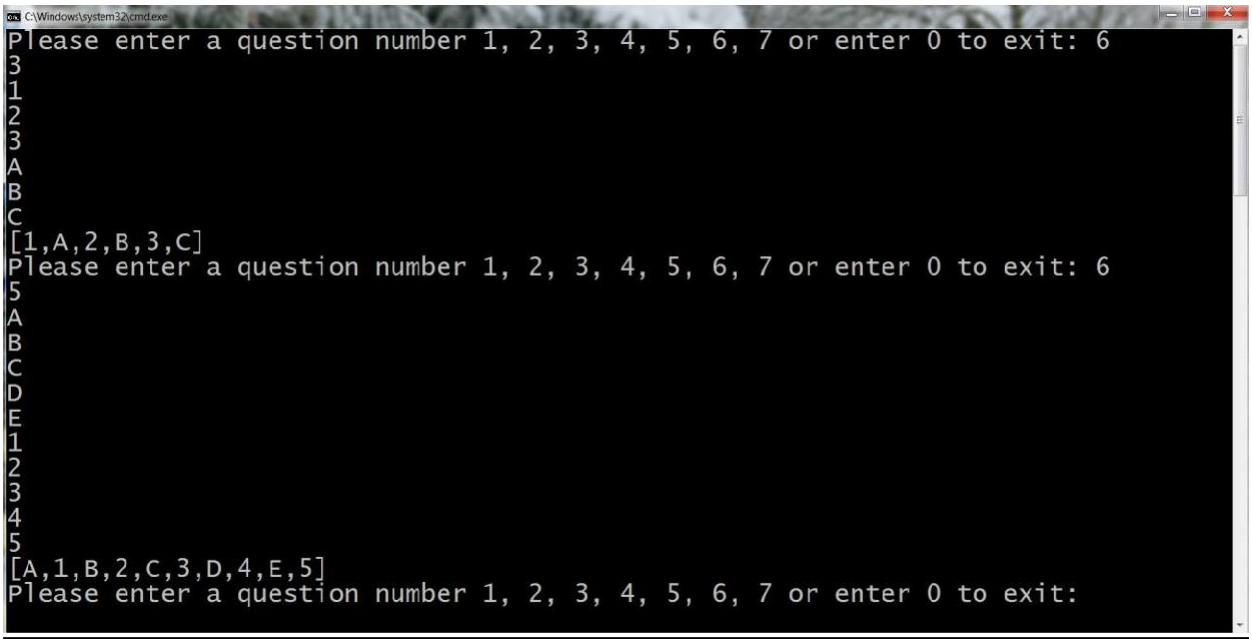

Figure 6 question-6 sample run

Q7. In the procedure called "Q7", Write a program to find the sum of the series 1 +11 + 111 + 1111 + .. n terms.

Constraints:

- All input numbers are in decimal.
- Input number range: $1 \leq N \leq 10$.

Sample Input:

5

8

10

Sample Output:

12345

12345678

1234567900

Sample Run:

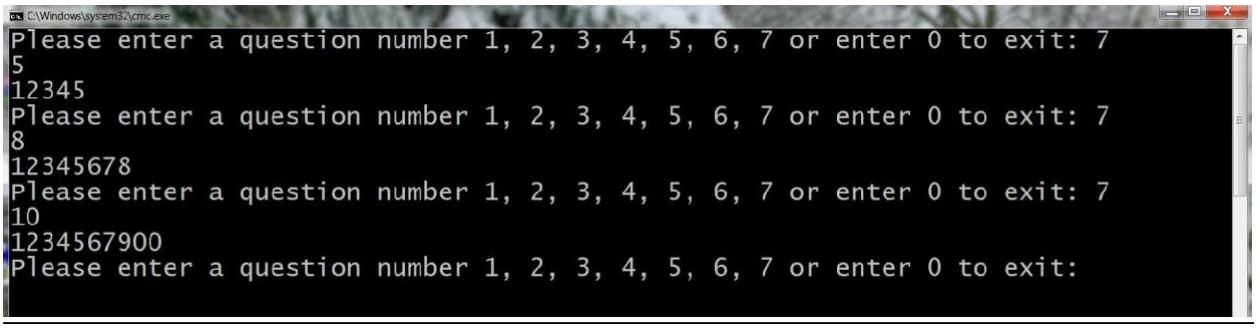

Figure 7 question-7 sample run

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.