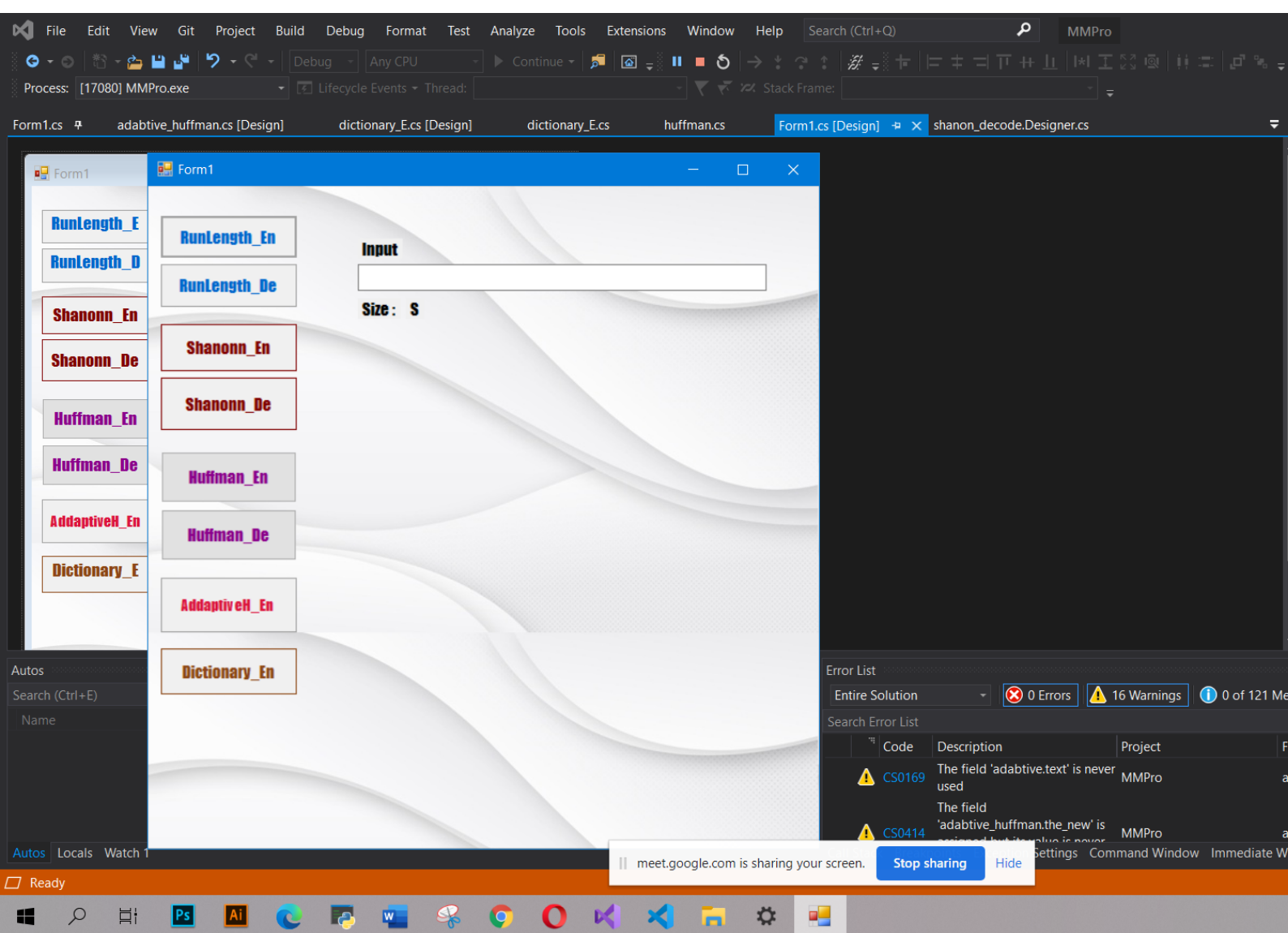


# Multimedia project

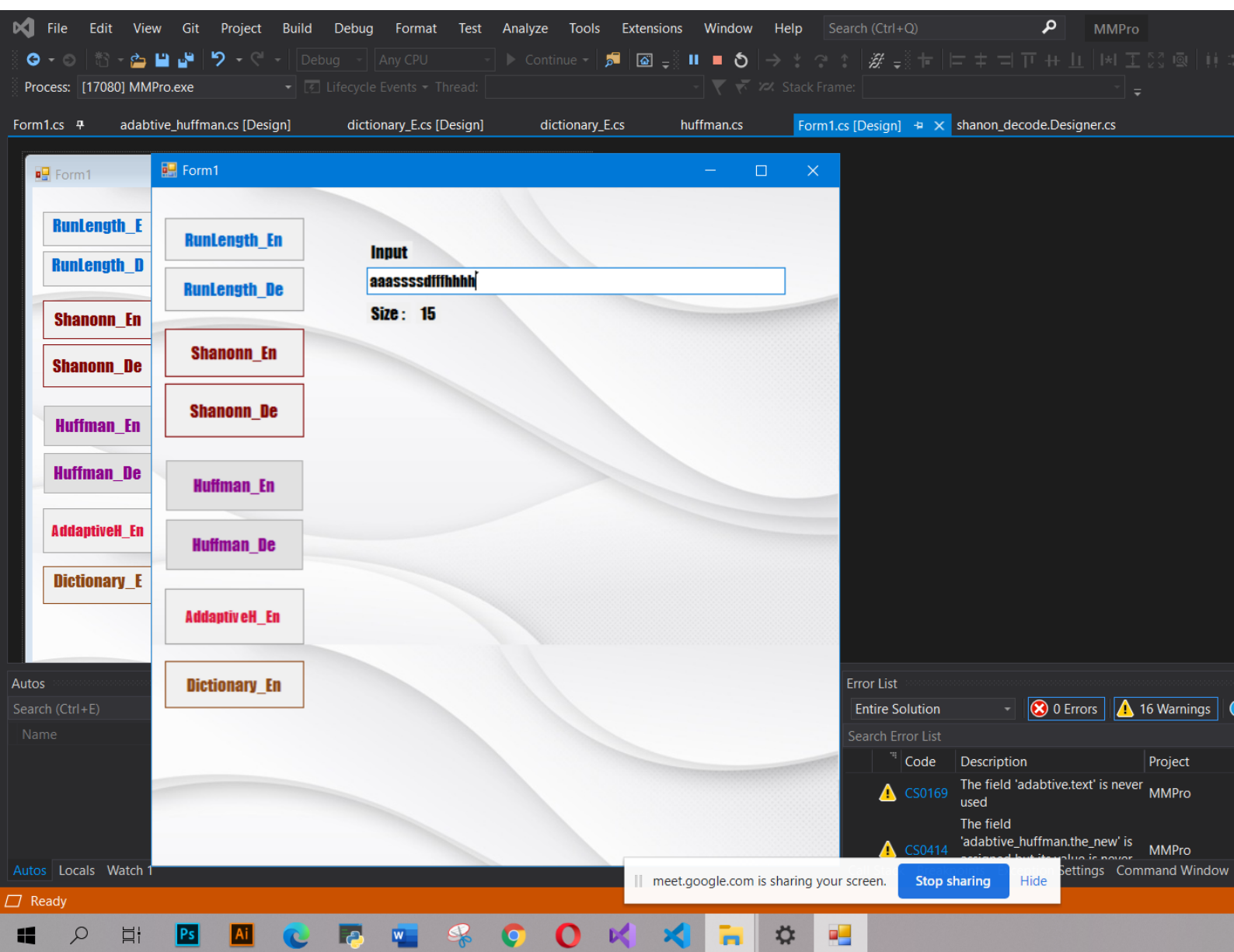
Made by: -

**Mona Ahmed Fouad Mansour – Sandra Atef**

- This the main windows form

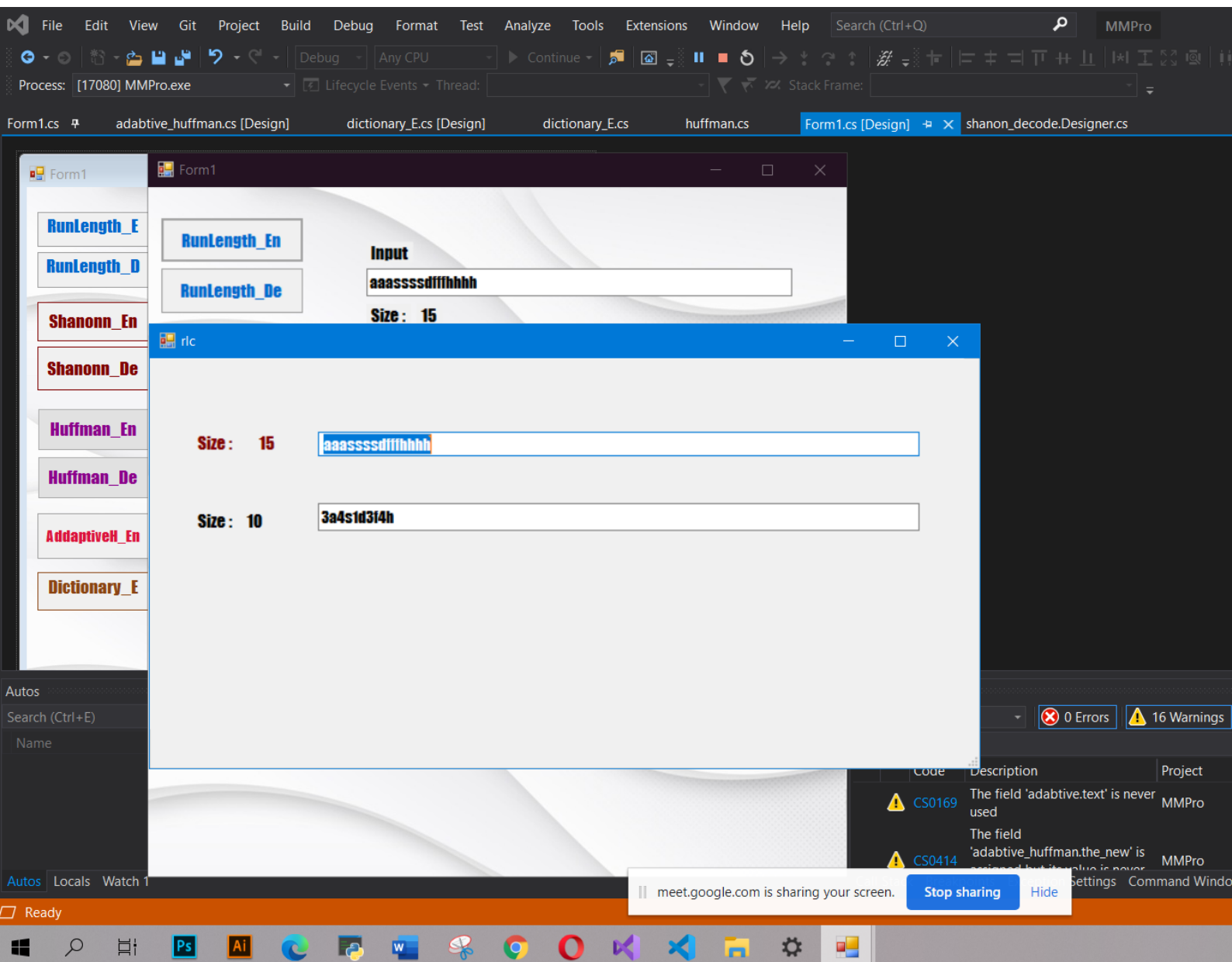


- Starts to put string inputs here



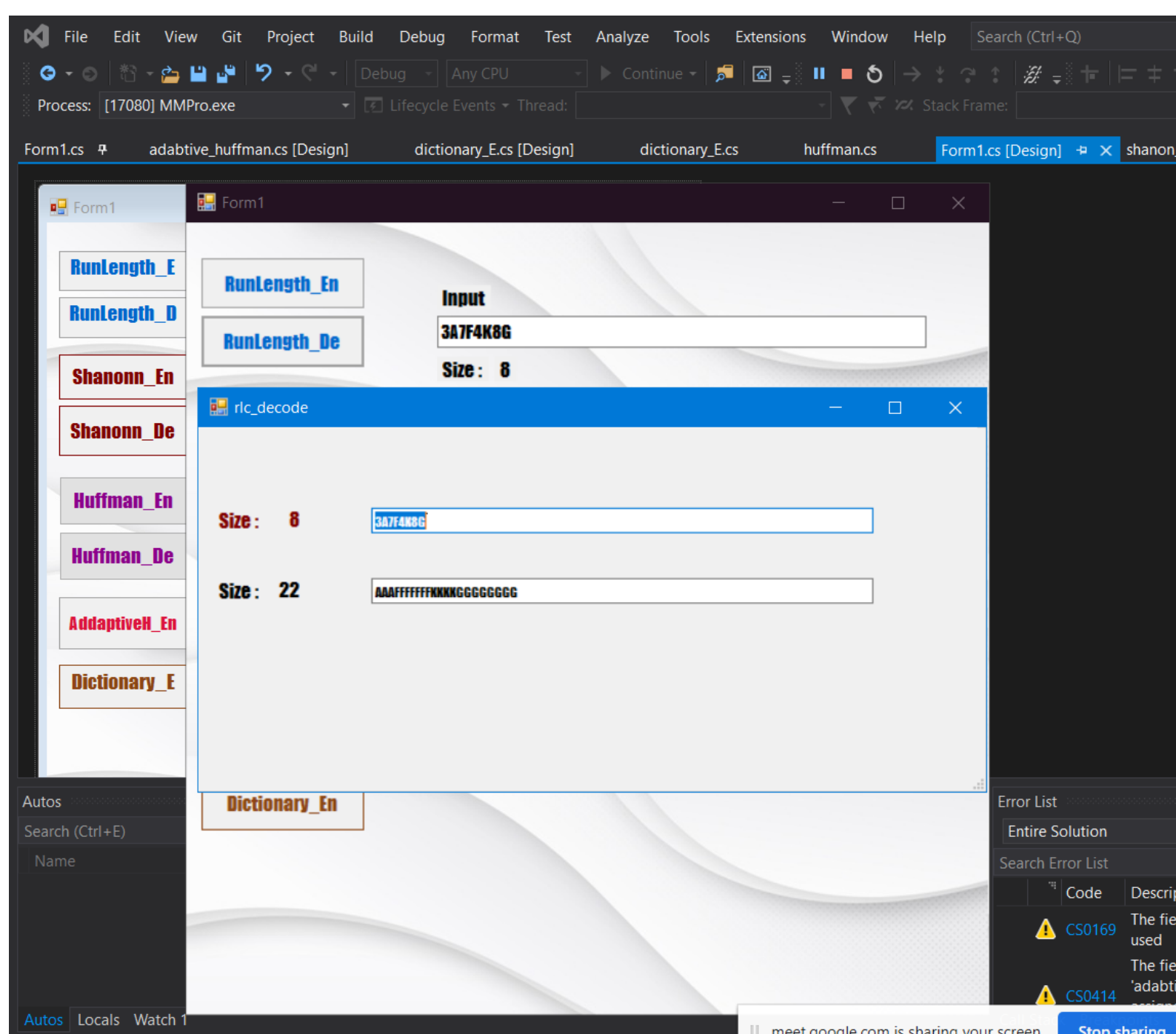
## Run length algorithm

Run-length encoding (RLE) is a form of lossless data compression in which runs of data (sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run. This is most useful on data that contains many such runs.



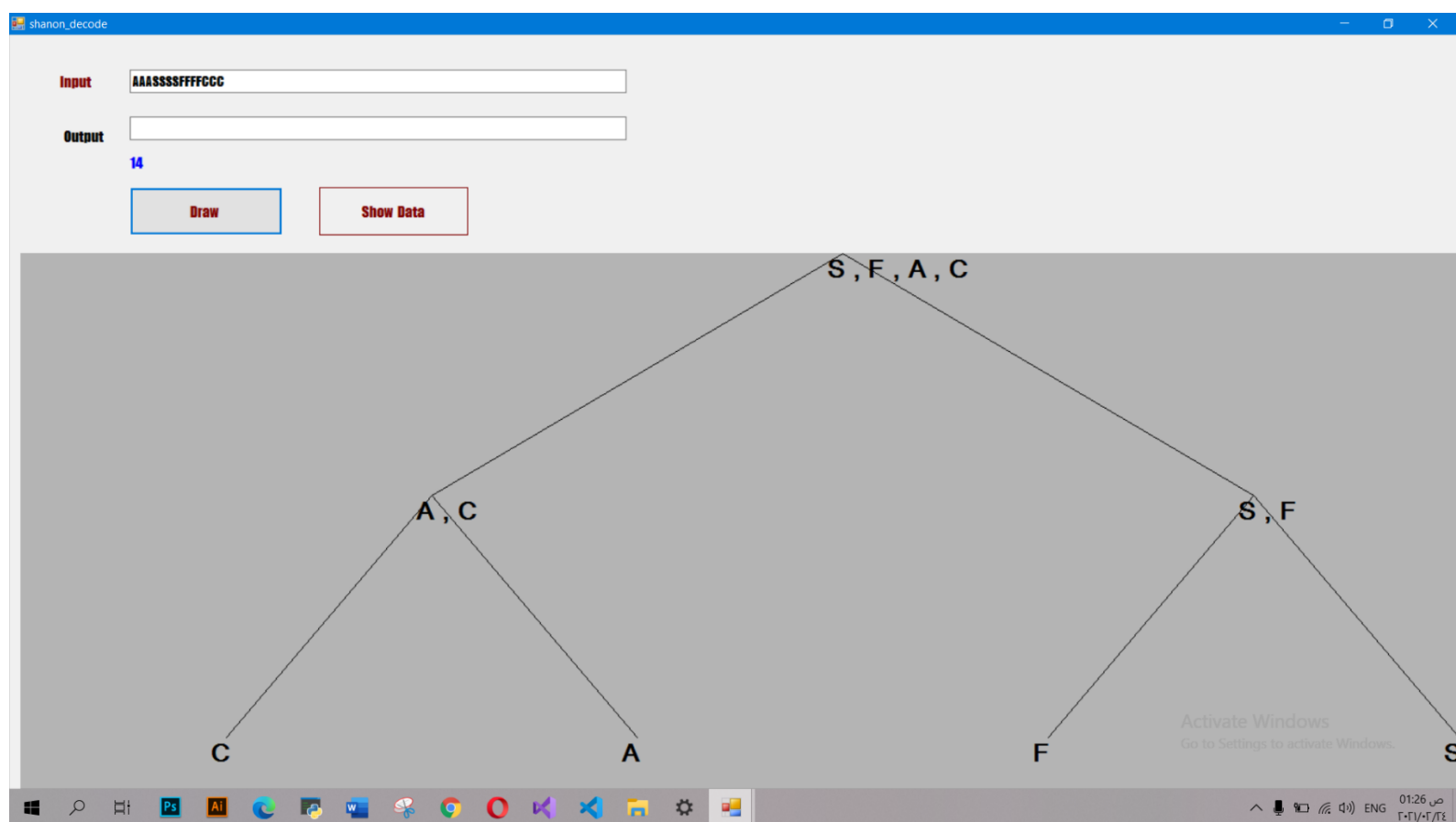
## Run length Decoding algorithm:

Run-length decoding (RLE) is a form of lossless data compression in which runs of data (sequences in which the same data value occurs in many consecutive data elements) are stored as an alphanumeric data and count, rather than as the original run. This is most useful on data that contains many such runs, in this window you put alphanumeric data compressed then the algorithm decompresses it



## Shannon algorithm:

is a lossless data compression technique for constructing a prefix code based on a set of symbols and their probabilities (estimated or measured) It is a variable length encoding scheme, that is, the codes assigned to the symbols will be of varying length, this window you put a string input and then starts to draw the tree and also you could show its bits like shown in data show form.



shanon\_decode

Input

AAAASSSSFFFFCCG

Output

14

Draw

Show Data

data\_show

	Sympol	bits	counter
▶	Sympol	bits	counter
	C	00	1
	A	01	1
	F	10	1
*	S	11	1

C

A

S, F, A, C

S, F

F

S

Activate Windows  
Go to Settings to activate Windows.

Windows icons

System tray

## Huffman code

is a way to encode information using variable-length strings to represent symbols depending on how frequently they appear. The idea is that symbols that are used more frequently should be shorter while symbols that appear more rarely can be longer

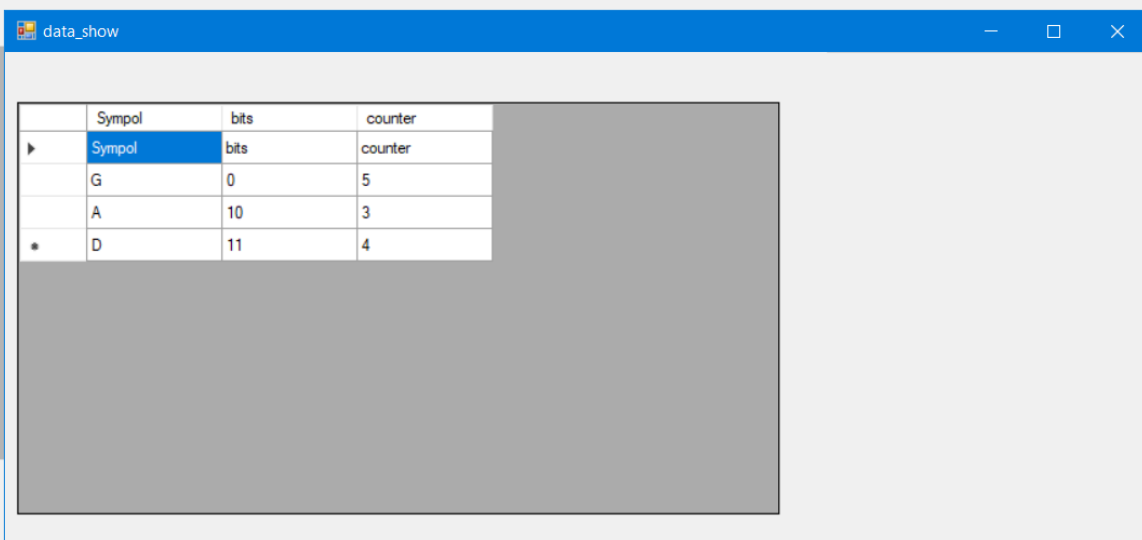
Huffman code assigns codes to characters such that the length of the code depends on the relative frequency or weight of the corresponding character

Input

AAADDDGGGGG

12

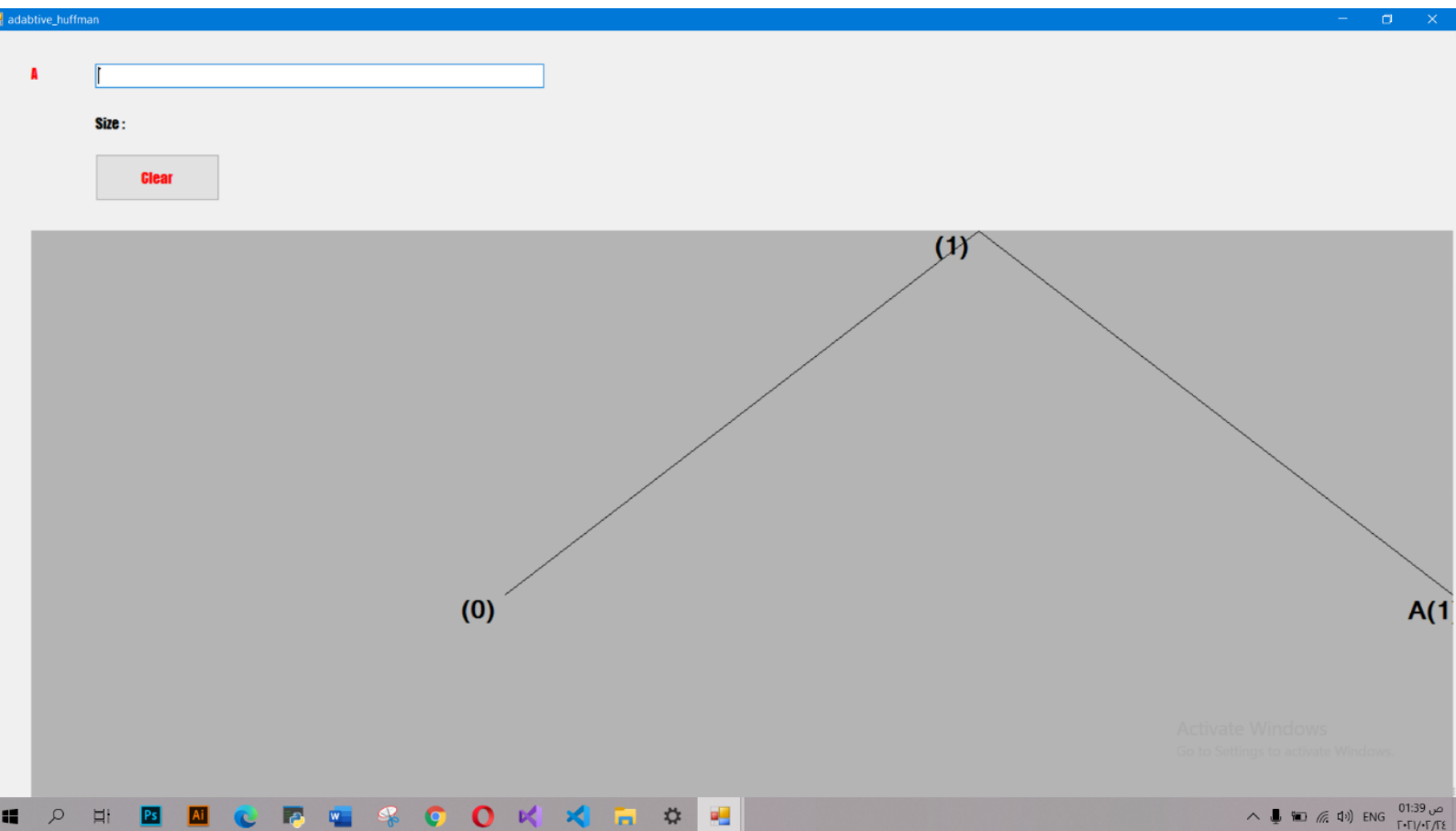
Data Show



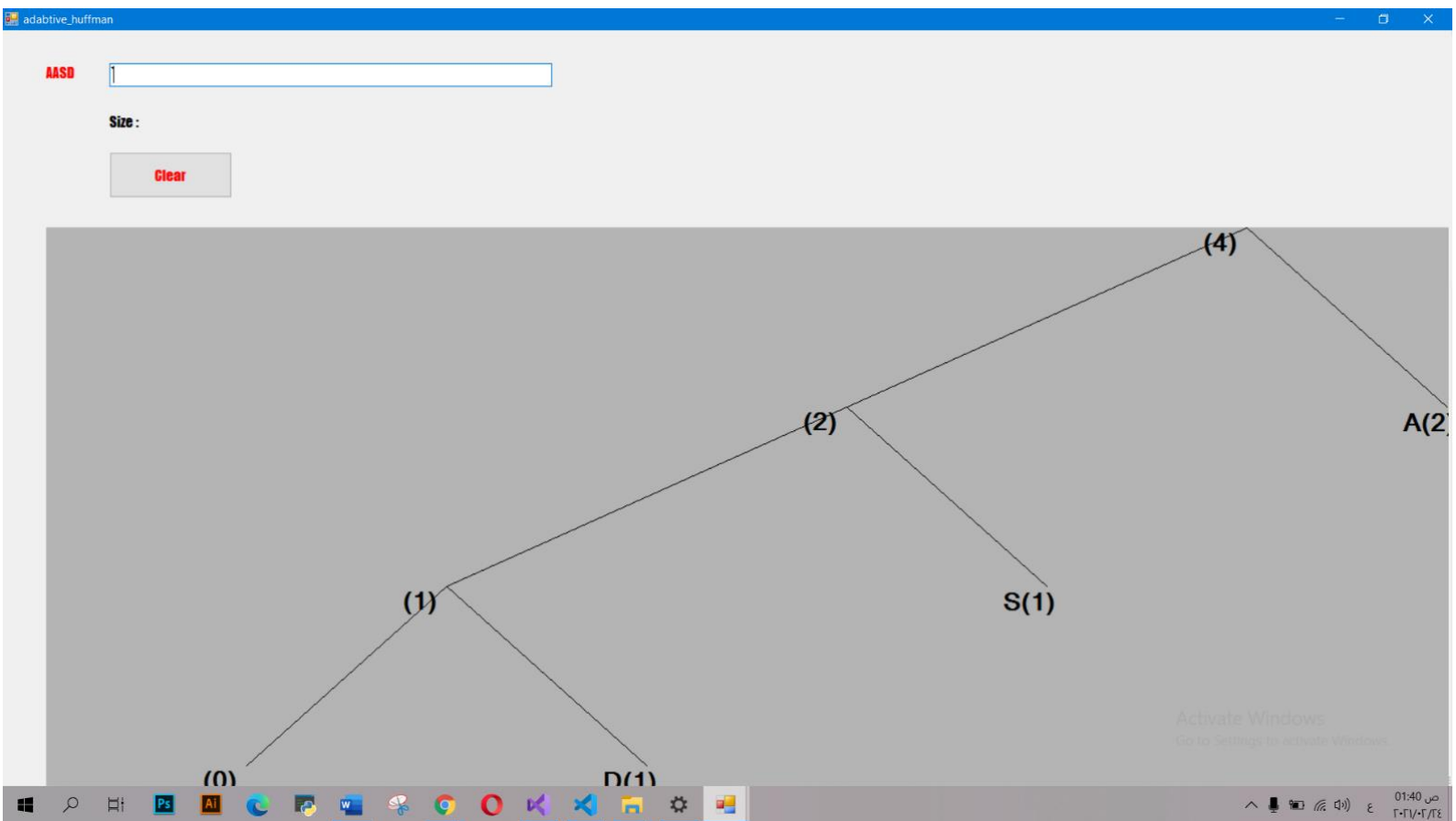
	Symbol	bits	counter
▶	Symbol	bits	counter
	G	0	5
	A	10	3
*	D	11	4

## Adaptive Huffman algorithm

is an adaptive coding technique based on Huffman coding. It permits building the code as the symbols are being transmitted, having no initial knowledge of source distribution, that allows one-pass encoding and adaptation to changing conditions in data, . In adaptive Huffman coding, the character will be inserted at the highest leaf possible to be decoded, before eventually getting pushed down the tree by higher-frequency characters. And in this window when you starts write a string the draw optimistically drawn



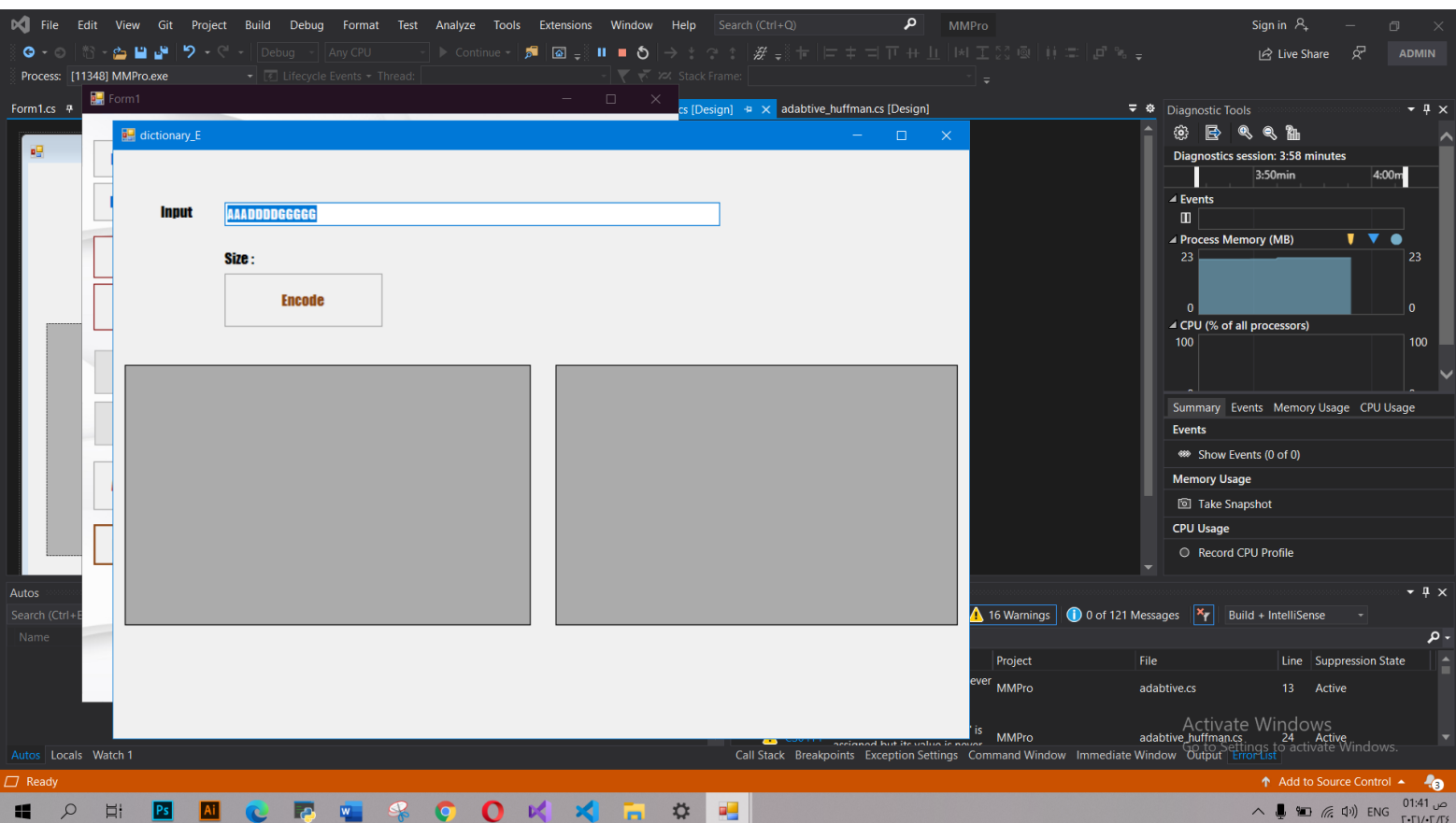




## Lzw

is a class of lossless data compression algorithms which operate by searching for matches between the text to be compressed and a set of strings contained in a data structure (called the 'dictionary') maintained by the encoder. When the encoder finds such a match, it substitutes a reference to the string's position in the data structure

The LZW algorithm is a greedy algorithm in that it tries to recognize increasingly longer and longer phrases that are repetitive, and encode them.



### Input

AAADDDGGGGG

**Size :**

## Encode

	Sympole	From	To	Range
►	A	0	0.25	0.25
	D	0.25	0.58333333...	0.33333333...
*	G	0.58333333...	1	0.41666666...

	Sympole	From	Range	
►	A	0	0.25	0.25
	D	0.0625	0.145833333...	0.08333333...
*	G	0.0486111111...	0.083333333...	0.03472222...