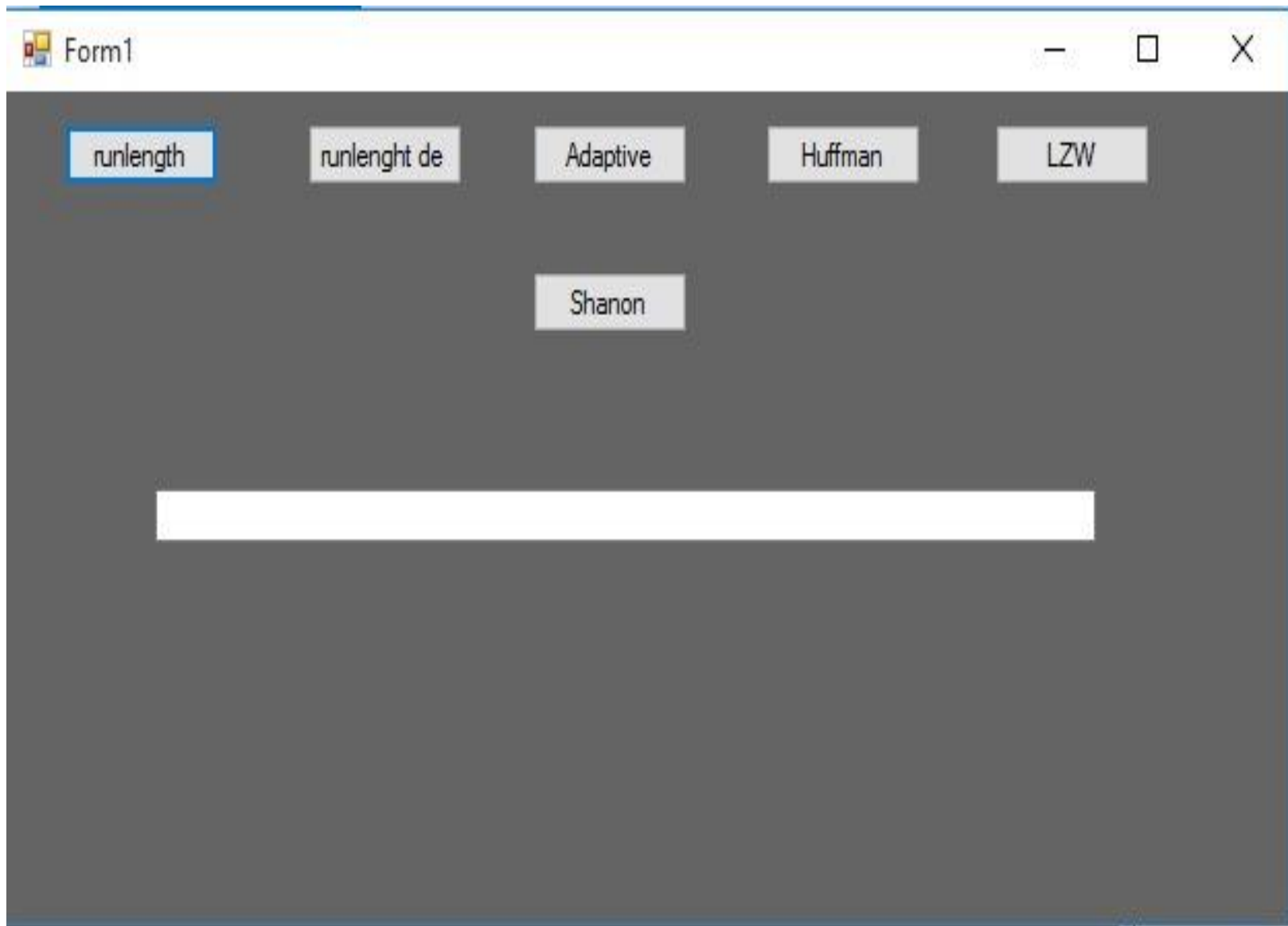


الاسم :جینا عاطف توفیق
سانی کی کمال شاهر



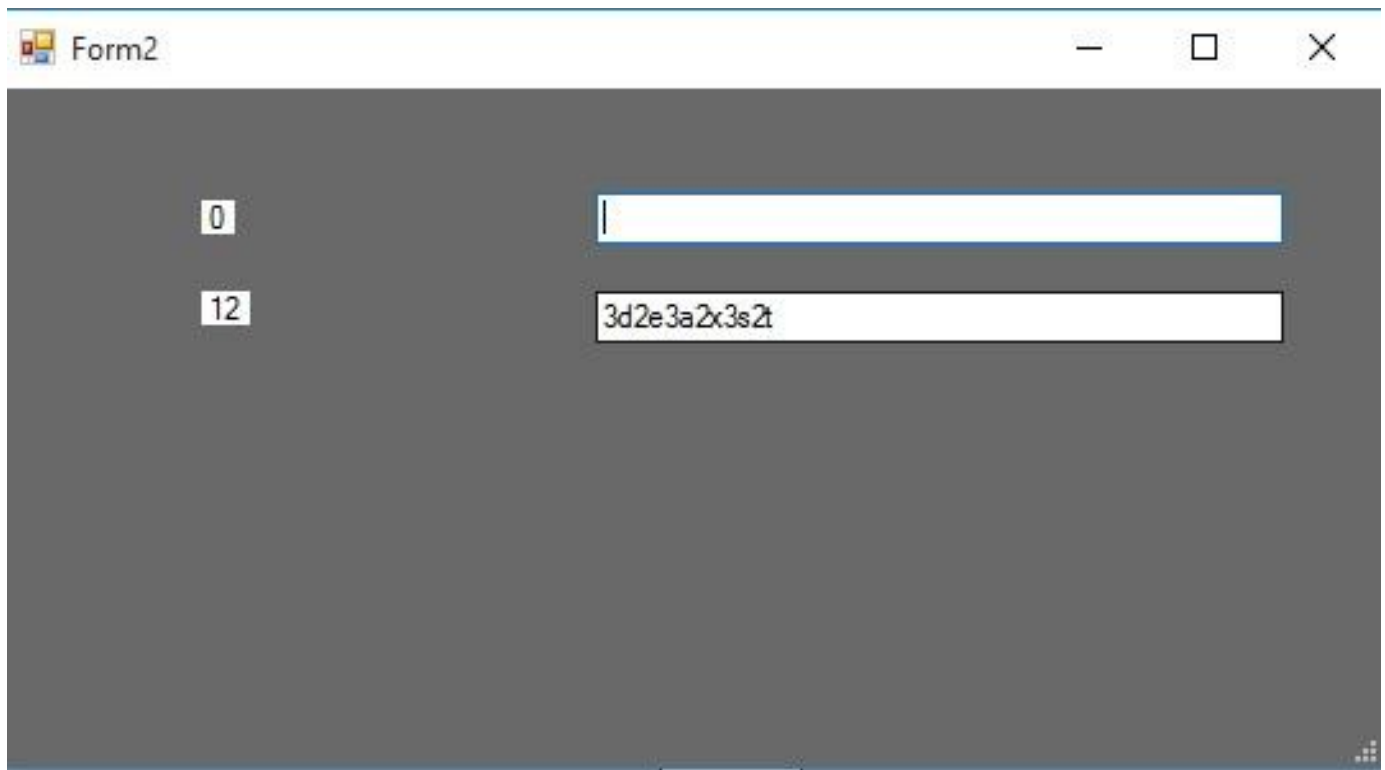
The image shows a screenshot of a Windows application window titled "Form1". The window has a standard Windows title bar with minimize, maximize, and close buttons. The main area of the window is dark gray and contains several buttons for different compression algorithms. The buttons are arranged in two rows. The first row contains five buttons: "runlength" (highlighted with a blue border), "runlenght de", "Adaptive", "Huffman", and "LZW". The second row contains one button: "Shanon". Below the buttons, there is a large white rectangular area, likely a text input field or a display area for the results of the compression process.

1.Run length compression:

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.

Consider, for example, simple graphic images such as icons, line drawings and animations. It is not useful with files that don't have many runs as it could greatly increase the file size.

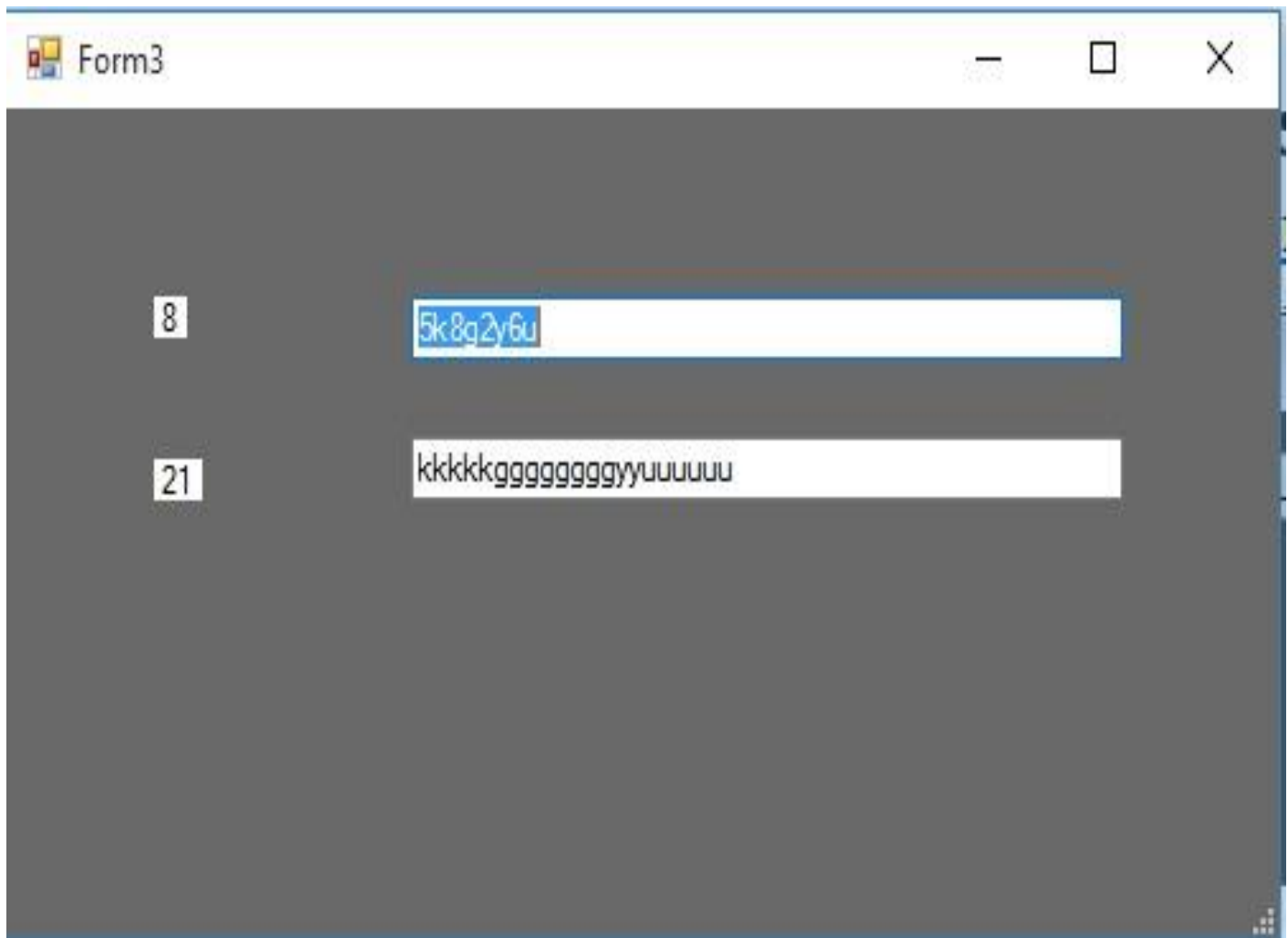


Form2

0	
12	3d2e3a2x3s2t

Run length Decoding:

Given an encoded Linked List which is encoded by” [Run Length Encoding](#) “ algorithm. The task is to decode the given linked list and generate the input string.

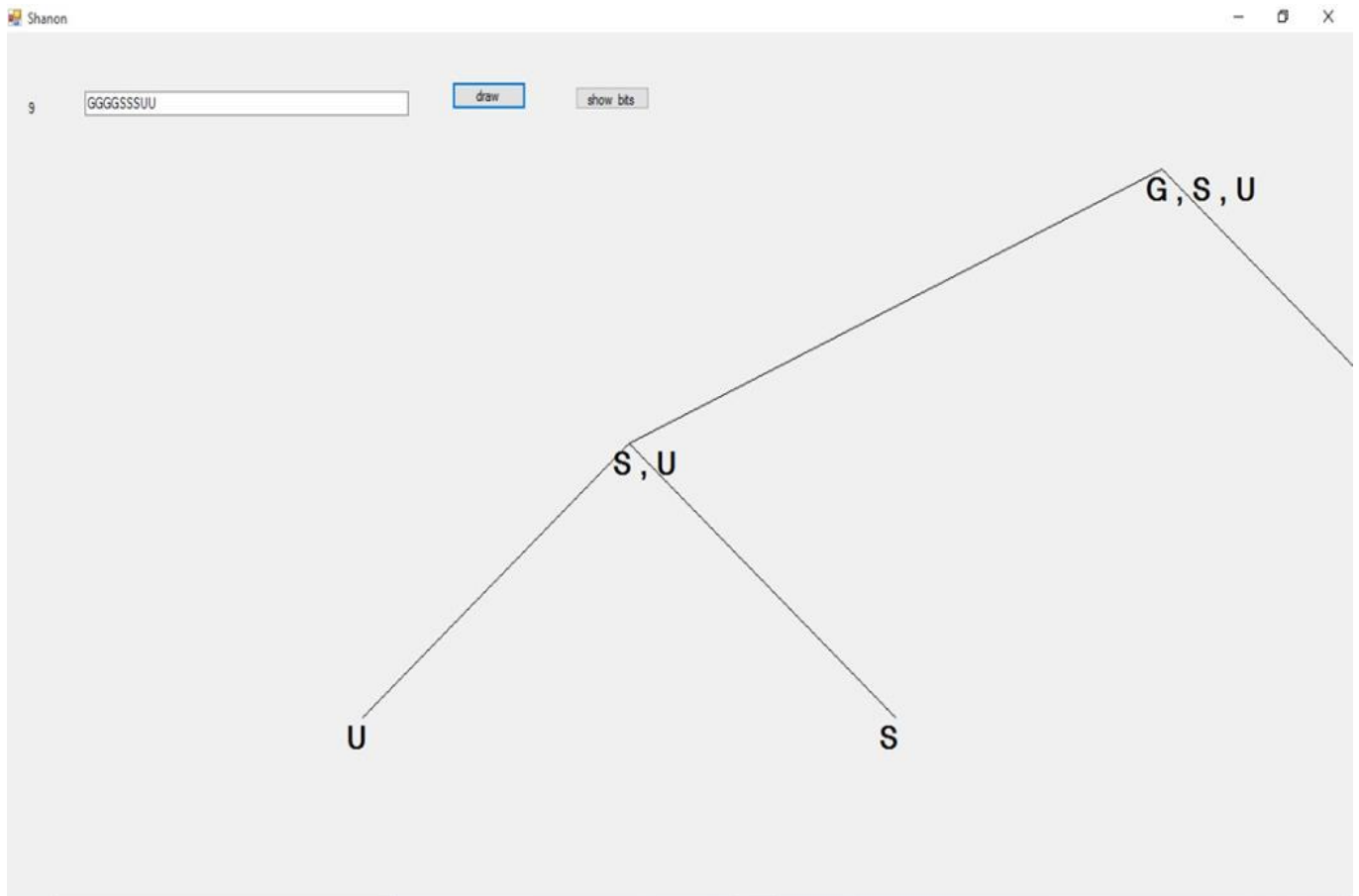


The screenshot shows a Windows application window titled "Form3". Inside the window, there are two rows of input fields. The first row has a small box containing the number "8" and a text box containing the string "5k8g2y6u". The second row has a small box containing the number "21" and a text box containing the string "kkkkkgggggggggyyyyyuuuuuu".

8	5k8g2y6u
21	kkkkkgggggggggyyyyyuuuuuu

2. Huffman coding:

Huffman code is a way to **encode information** using variable-length strings to represent symbols depending on how frequently they appear. **Huffman coding** uses a greedy algorithm to build a prefix **tree** that optimizes the **encoding** scheme so that the most frequently used symbols have the shortest **encoding**.



3. Variable length coding:

In [coding theory](#) a **variable-length code** is a [code](#) which maps source symbols to a *variable* number of bits.

Variable-length codes can allow sources to be [compressed](#) and decompressed with *zero* error ([lossless data compression](#)) and still be read back symbol by symbol.

	Sympol	bits	counter
►	Sympol	bits	counter
	f	0	5
	g	10	3
*	s	11	4

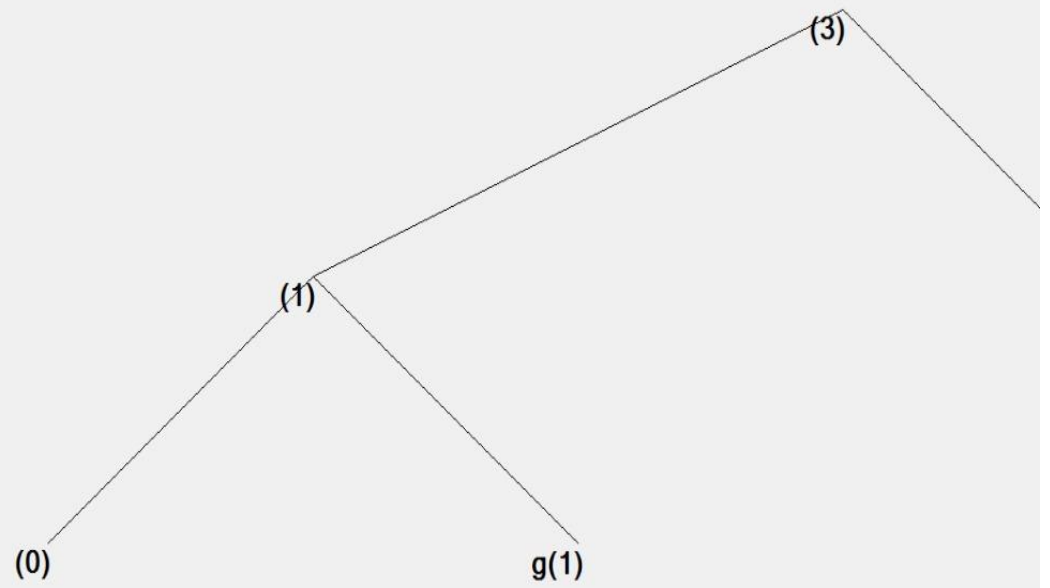
4. Adaptive Huffman coding:

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. It is an online coding technique based on Huffman coding. Having no initial knowledge of occurrence frequencies, it permits dynamically adjusting the Huffman's tree..

	Sympol	bits	counter
►	Sympol	bits	counter
	f	0	5
	g	10	3
*	s	11	4

gff

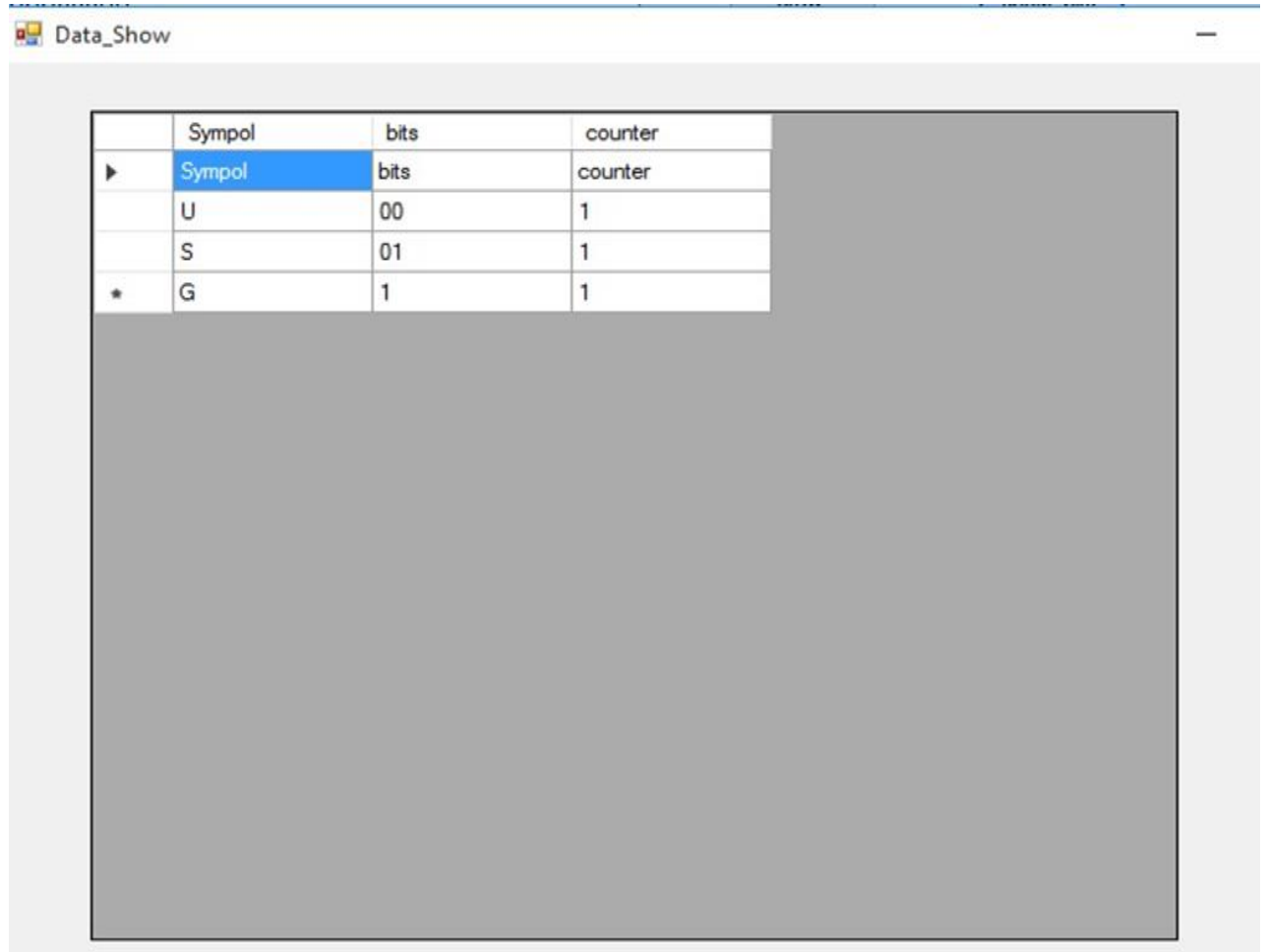
clear



5. Shannon Fano encoding:

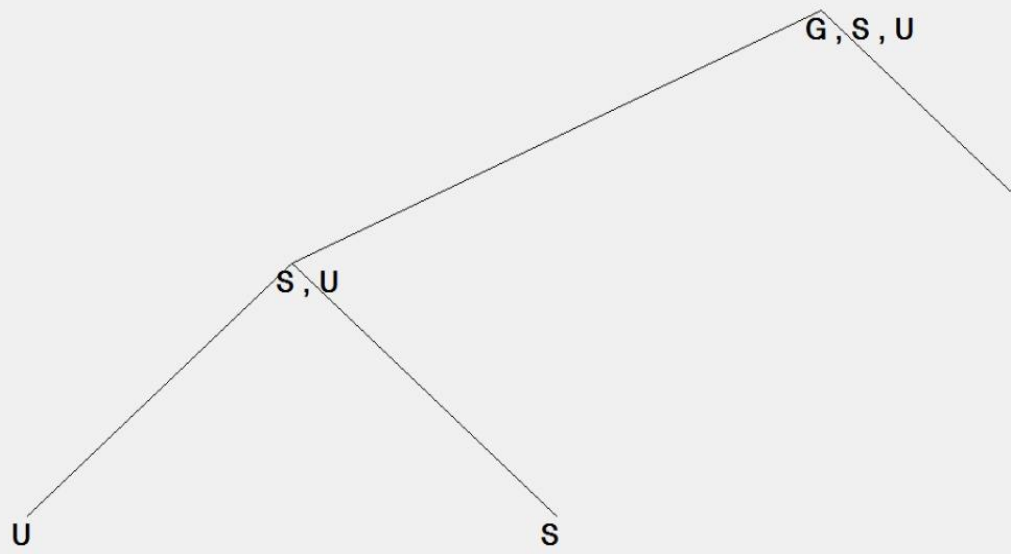
In the field of [data compression](#), **Shannon–Fano coding**, named after [Claude Shannon](#) and [Robert Fano](#), is a name given to two different but related techniques for constructing a [prefix code](#) based on a set of symbols and their probabilities (estimated or measured).

- **Fano's method** divides the source symbols into two sets ("0" and "1") with probabilities as close to 1/2 as possible. Then those sets are themselves divided in two, and so on, until each set contains only one symbol. The code word for that symbol is the string of "0"s and "1"s that records which half of the divides it fell on.



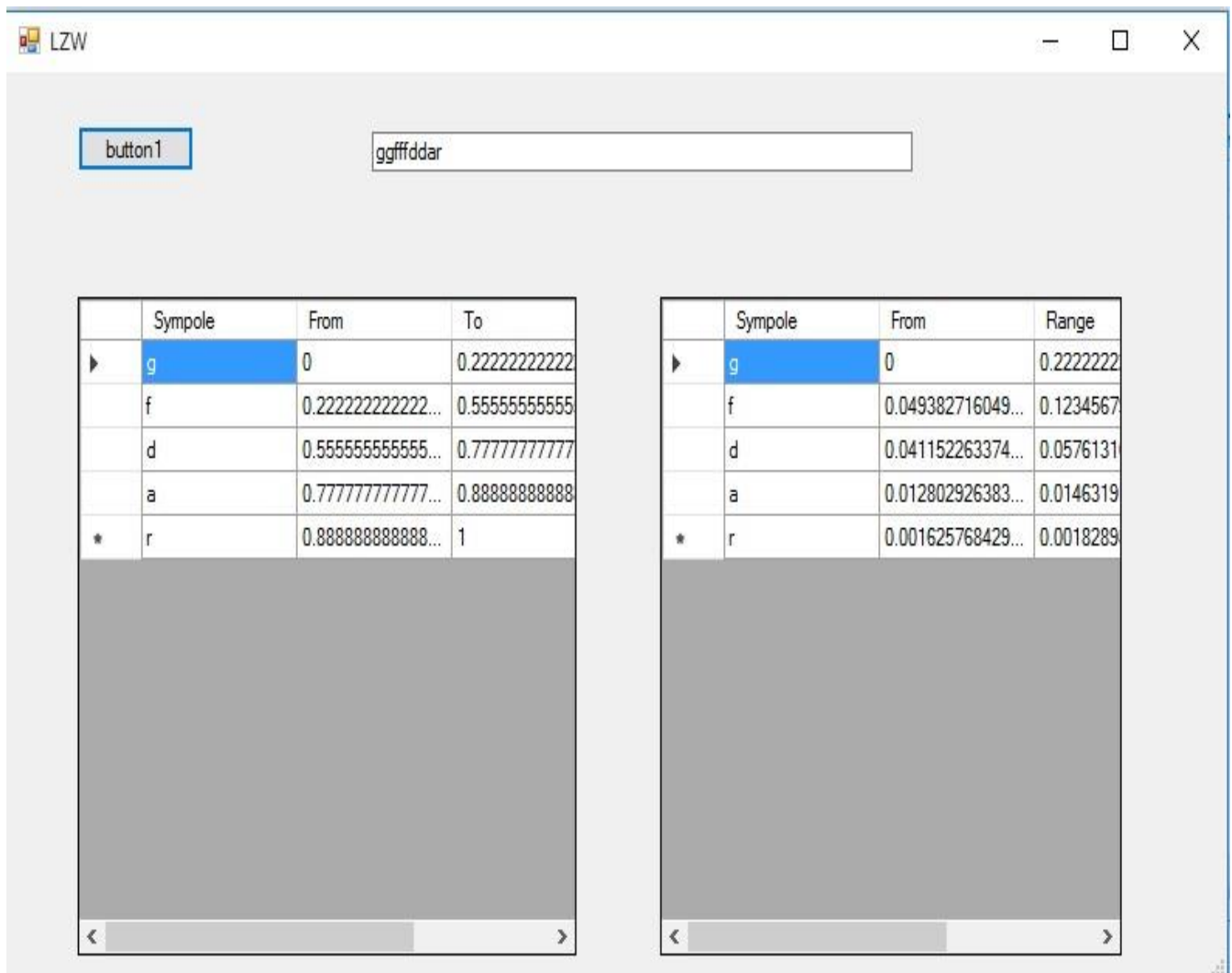
	Sympol	bits	counter
►	Sympol	bits	counter
	U	00	1
	S	01	1
*	G	1	1

g



6. LZW (Dictionary):

Is a universal lossless data **compression algorithm** created by Abraham Lempel, Jacob Siva, and Terry Welch? ... It is the **algorithm** of the widely used UNIX file **compression** utility compress and is used in the GIF image format. It also sometimes known as a **substitution coder**, 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 screenshot shows a window titled "LZW" with a button labeled "button1" and a text input field containing "ggffddar". Below the input field are two side-by-side tables representing dictionaries. The left table has columns "Symbole", "From", and "To". The right table has columns "Symbole", "From", and "Range". Both tables show a sequence of characters and their corresponding numerical values or ranges.

	Symbole	From	To
▶	g	0	0.2222222222
	f	0.2222222222...	0.5555555555
	d	0.5555555555...	0.7777777777
	a	0.7777777777...	0.8888888888
*	r	0.8888888888...	1

	Symbole	From	Range
▶	g	0	0.2222222
	f	0.049382716049...	0.1234567
	d	0.041152263374...	0.0576131
	a	0.012802926383...	0.0146319
*	r	0.001625768429...	0.0018289

