Notes/answers to the tutorial questions:

Huffman coding:

1. The given probabilities result in a linear tree. Normally, when the probability has such a regularity: P(i+1) = P(i) + d (where i represents the index of a character in the alphabet, d is a constant), a linear tree will be generated.
2. You can construct many different distributions that could lead to such huffman trees. Ensure that you not only consider the difference between the probabilities of these characters themselves, but also carefully design them to make their summation to satisfy the given Huffman trees.
3. Normally, Huffman trees with unbalanced nodes will give more benefits considering the compression rate.

LempelZiv:

1. Everytime when there are multiple occurrence, just use the first occurrence in the sliding window.
2. LZ method cannot generate a good compression rate for the given sequence. Normally, LZ is good at encoding long text written in natural languages or images/videos where a lot of repetitions are expected.
3. They are supposed to be generating the same string “to_be_or_not_to_be”. But the second one is incorrect.

String Search:

1. The partial matching table for S:

A	B	C	D	A	B	D
-1	0	0	0	0	1	2

2. The partial matching table:

A	**B**	**A**	**B**	**A**	**C**	**A**	**B**	**A**	**…**
-1	**0**	**0**	**1**	**2**	**3**	**0**	**1**	**2**	**3**