Question 1
Suppose the alphabet is [A,B,C], and the known probability distribution is $P_A = 0.5$; $P_B = 0.4$; $P_C = 0.1$. For simplicity, let us assume that both the encoder and decoder know that the length of the messages is always 5, so there is no need for a terminator.

1. How many bits are needed to encode the message AABCC by Huffman coding? Illustrate with Huffman trees.
2. How many bits are needed to encode the message AABCC by Arithmetic Coding?

Question 2
Consider the dictionary-based LZW compression algorithm. Suppose the alphabet is the set of symbols {p, q}. Show the dictionary (Symbol sets plus associated codes) and output for LZW compression of the input:

p q q p p q q

Question 3

Design the adaptive Huffman coding with initial code assignment for the string **AAD-CCDDDDB** as follows

New	A	B	C	D
0	001	010	011	100

If any character is to be sent the first time, it must be preceded by a special symbol, NEW. The initial code for NEW is 0. The count for NEW is always kept as 0 (the count is never increased);

1. What is the entropy η of the string?
2. Draw the adaptive Huffman trees which are built after sending each character.
3. Write down the codes sent to the decoder after each step.

4. Are the following procedures for adaptive Huffman coding correct or wrong? Correct the algorithm if needed.

```
ENCODER                 DECODER

Initial_code()          Initial_code()
while not EOF           while not EOF
  {                       {
    get(c)                  update_tree
    encode(c)               decode(c)
    update_tree             output(c)
  }                       }
```

5. Apply the LZW compression on the string with the initial dictionary showing in table below:

code	String
1	A
2	B
3	C
4	D

What is the output code and what is the compression ratio? Assume each character or code is transmitted as a byte.

——— *End of paper* ———

Appendix A: Equation List

The entropy:

$$\eta = \sum p_i \log_2 \frac{1}{p_i} \tag{1}$$

Multinomial:

$$P(c|d) \;\propto\; P(c)\Pi_{1\leqslant k\leqslant N_d} P(t_k|c) \tag{2}$$

$$P(t_k|c) \;=\; \frac{T_{ct}+1}{\sum_{t'\in V(T_{ct'}+1)}} \tag{3}$$

Bernoulli:

$$P(c|d) \;\propto\; P(c)\Pi_{t_k\in Q} P(t_k|c)\Pi_{t_k\notin Q}[1-P(t_k|c)] \tag{4}$$

$$P(t_k|c) \;=\; \frac{df_{ct}+1}{N_c + Numerofclasses} \tag{5}$$

Arithmetic Coding Encoder:

```
BEGIN
   low = 0.0;    high = 1.0;    range = 1.0;

   while (symbol != terminator)
       {
         get (symbol);
         low =  low + range * Range_low(symbol);
         high = low + range * Range_high(symbol);
         range = high - low;
       }

   output a code so that low <= code < high;
END
```

ALGORITHM 7.2 LZW COMPRESSION

```
BEGIN
   s = next input character;
   while not EOF
        {
          c = next input character;

          if s + c exists in the dictionary
             s = s + c;
      else
         {
         output the code for s;
         add string s + c to the dictionary with a new code;
         s = c;
         }
      }
   output the code for s;
END
```