Problem 4.

a)

$$H = \sum_{i=1}^{5} P(a_i) \log P(a_i) = 1.817684 \text{bits}$$

b)

If we sort the probabilities in descending order, we can see that the two letters with the lowest probabilities are a_2 and a_4. These will become the leaves on the lowest level of the binary tree. The parent node of these leaves will have a probability of 0.9. If we consider parent node as a letter in a reduced alphabet then it will be one of the two letters with the lowest probability: the other one being a_1. Continuing in this manner, we get the binary tree shown in Figure 1. and the code is

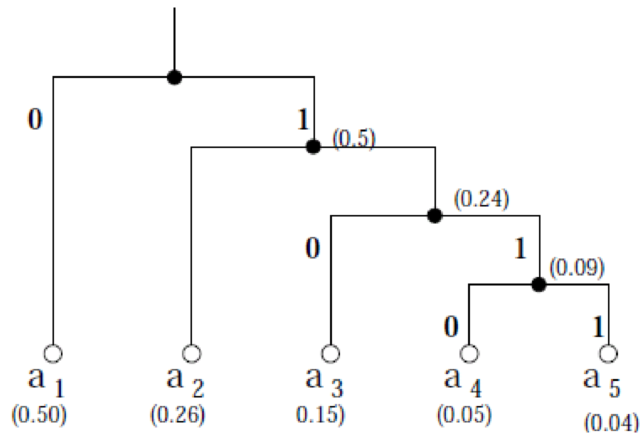

Figure 1: Huffman code for the five-letter alphabet.

a_1	110
a_2	1111
a_3	10
a_4	1110
a_5	0

c) $l_{avg} = 0.15 \times 3 + 0.04 \times 4 + 0.26 \times 2 + 0.05 \times 4 + 0.5 \times 1 = 1.83$bits/symbol.

Problem 5.

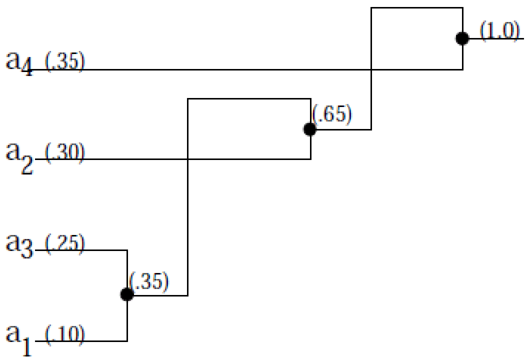

Figure 2: Huffman code for the four-letter alphabet in Problem 5.

a) The Huffman code tree is shown in Figure 2. The code is

a_1	011
a_2	01
a_3	010
a_4	1

The average length of the code is 0.1×3+0.3×2+0.25×3+0.35×1 = 2 bits/symbol.

b) Huffman code tree is shown in Figure 3. The code is

a_1	01
a_2	11
a_3	00
a_4	10

The average length of the code is obviously 2 bits/symbol.

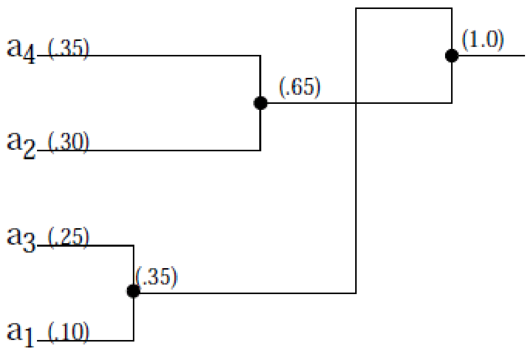

Figure 3: Minimum variance Huffman code for the four-letter alphabet in Problem 5.

While the average length of the codeword is the same for both codes, that is they are both equally efficient in terms of rate. However, the second code has a variance of zero for the code lengths. This means that we would not have any problems with buffer control if we were using this code in a communication system. We cannot make the same assertion about the first code.

Problem 6.

Examining the Huffman code generated in Problem 4 (not 3!) along with the associated probabilities, we have

a_1	110	0.15
a_2	1111	0.04
a_3	10	0.26
a_4	1110	0.05
a_5	0	0.50

The proportion of zeros in a given sequence can be obtained by first computing the probability of observing a zero in a codeword $\sum_{k=1}^{5} P(0|a_k)P(a_k)$ and then dividing that by the average length of a codeword. The probability of observing a zero in a codeword is

$$1 \times 0.15 + 0 \times 0.04 + 1 \times 0.26 + 1 \times 0.05 + 1 \times 0.50 = 0.96.$$

0.96/1.83 = 0.52. Thus, the proportion of zeros is close to a half. If we examine Huffman codes for sources with dyadic probabilities, we would find that the proportion is exactly a half. Thus, the use of a Huffman code will not lead to inefficient channel usage.

Problem 10.

a)

Message	$a_2a_1a_3a_2a_1a_2$
Transmitted binary sequence	1010001011
Received binary sequence	0010001011
Decoded sequence	$a_4a_4a_2a_2$

Depending on how you count, the errors five characters are received in error before the first correctly decoded character.

b)

Transmitted binary sequence	001011001000
Received binary sequence	101011001000
Decoded sequence	$a_1a_1a_3a_2a_1a_2$

Only a single character is received in error.

c)

Message	$a_2a_1a_3a_2a_1a_2$
Transmitted binary sequence	1010001011
Received binary sequence	1000001011
Decoded sequence	$a_2a_3a_4a_2a_2$

four characters are received in error before the first correct character.

For the minimum variance code the situation is different

Message	$a_2a_1a_3a_2a_1a_2$
Transmitted binary sequence	001011001000
Received binary sequence	000011001000
Decoded sequence	$a_2a_2a_3a_2a_1a_2$

Again, only a single character is received in error.

Problem 13.

First iteration:

Letter	Probability
a_1	0.7
a_2	0.2
a_3	0.1

Second iteration:

Letter	Probability
a_2	0.2
a_3	0.1
a_1a_1	0.49
a_1a_2	0.14
a_1a_3	0.07

Final iteration:

Letter	Code
a_2	000
a_3	001
a_1a_2	010
a_1a_3	011
$a_1a_1a_1$	100
$a_1a_1a_2$	101
$a_1a_1a_3$	110