1 Problem 1

For a crypto system to have perfect secrecy, the cipher text should not reveal any information about the plain text. That is,

$$Pr[p|c] = Pr[p]$$  \hspace{1cm} (1)

Using bayes theorem the following must be true

$$Pr[p|c] = \frac{Pr[c|p]Pr[p]}{Pr[c]}$$  \hspace{1cm} (2)

We know that $Pr[c|p] = \frac{1}{n}$. This is because a latin square has exactly one of every $n$ in a row in the latin square. That is, given any plaintext, it could be any ciphertext, all with equally likely values of $\frac{1}{n}$.

This is also true for $Pr[c]$. For $n^2$ possible values of ciphertexts, there are $n$ that corresponding to the one in question.

Therefore,

$$Pr[p|c] = \frac{\frac{1}{n}Pr[p]}{\frac{1}{n}}$$  \hspace{1cm} (3)

$$Pr[p|c] = Pr[p]$$  \hspace{1cm} (4)

2 Problem 2

2.1 Part A

In a set of $n$ values, if all $n$ values have equally likely probability of occurring $Pr[n] = \frac{1}{n}$, then a huffman tree created to encode these $n$ values will be a complete (but not balanced) binary tree.
This means that $2^{k+1} - n$ values have length $k$, and the remaining $n - (2^{k+1} - n)$ values are one level deeper on the huffman tree and have length $k + 1$.

If we are to solve for $\ell(f)$, we can find the weighted average of the lengths.

$$\ell(f) = \frac{(2^{k+1} - n)(k) + (2n - 2^{k+1})(k + 1)}{n} \quad (5)$$

$$\ell(f) = \frac{(k2^{k+1} - kn) + (2kn - k2^{k+1} + 2n - 2^{k+1})}{n} \quad (6)$$

$$\ell(f) = \frac{kn + 2n - 2^{k+1}}{n} \quad (7)$$

$$\ell(f) = n + 2 - \frac{2^{k+1}}{n} \quad (8)$$

### 2.2 Part B

If $n = 6$, then $k$ must be 2. This is because $k = 2$ satisfies the inequality: $2^2 \leq 6 < 2^3$.

If we know $k$, we can easily find $\ell(f)$ and $H(X)$. Let’s start with $\ell(f)$.

$$\ell(f) = 2 + 2 - \frac{8}{6} \quad (9)$$

$$\ell(f) = 2.67 \quad (10)$$

Now $H(X)$.

$$H(X) = \log_2(6) \quad (11)$$

$$H(X) = 2.58 \quad (12)$$

### 3 Problem 3

Given that $X = \{a, b, c, d, e\}$, we must build a prefix free binary encoding if $X$ using Huffman’s algorithm.

the probabilities of each letter in $X$ occurring is as follows.
\[ \Pr[a] = 0.32 \]
\[ \Pr[b] = 0.23 \]
\[ \Pr[c] = 0.20 \]
\[ \Pr[d] = 0.15 \]
\[ \Pr[e] = 0.10 \]

(13)

Given these probabilities, we can start to build the tree.

We first start with the two letters with the lowest probability of occurring. The two least occurring letters are \( e \) and \( d \). We assign these two letters an arbitrary value of either 0 or 1. We will set \( e \) to 0 and \( d \) to 1.

We now have one section of our huffman tree. We can combine the probabilities of \( e \) and \( d \) to form a new node, which we will call \( z_1 \), in our huffman tree.

Now we have a new set, with the following probabilities.

\[ \Pr[a] = 0.32 \]
\[ \Pr[b] = 0.23 \]
\[ \Pr[c] = 0.20 \]
\[ \Pr[z_1] = 0.25 \]

(14)

we can repeat the process above for the next two least likely letters, which are \( c \) and \( b \). Using these, we will form a new node, \( z_2 \). We will assign 0 to \( c \) and 1 to \( b \).
This brings us to yet another distribution, with the following probabilities.

\[
\begin{align*}
\Pr[a] &= 0.32 \\
\Pr[z_2] &= 0.43 \\
\Pr[z_1] &= 0.25
\end{align*}
\] (15)

Let’s repeat again, with \( z_1 \) and \( a \). We will create a new node, \( z_3 \) and assign 0 to \( z_1 \) and 1 to \( a \).

This leaves us with two nodes left, which we can combine into the root node and have a complete tree. We can assign 0 to \( z_2 \) and 1 to \( z_3 \).

Therefore our code is the following.

\[
\begin{align*}
a &= 11 \\
b &= 01 \\
c &= 00 \\
d &= 101 \\
e &= 100
\end{align*}
\] (16)
We can also now find $\ell(f)$ and $H(x)$.

$$\ell(f) = \sum_i \text{len}(n_i) \Pr[n_i]$$  \hspace{1cm} (17)

$$\ell(f) = 2(0.32) + 2(0.23) + 2(0.20) + 3(0.15) + 3(0.15)$$  \hspace{1cm} (18)

$$\ell(f) = 2.25$$  \hspace{1cm} (19)

and now $H(X)$.

$$H(X) = -\sum_i \Pr[n_i] \log_2(\Pr[n_i])$$  \hspace{1cm} (20)

$$H(X) = -[0.32\log_2(0.32) + 0.23\log_2(0.32) + ... + 0.10\log_2(0.10)]$$  \hspace{1cm} (21)

$$H(X) = 2.22$$  \hspace{1cm} (22)