Problem 1

Note that each letter of the genome sequence is independent and uniformly chosen from \{A, C, T, G\}

To show that 2 is equivalent to 2', we must verify that the conditional probability of the next letter given the previous letter is the same in both cases.

By symmetry, we only have to check the conditional probabilities \( p(A|A) \) and \( p(C|A) \).

Let \( M \) be the event that there is a mutation. Then \( p(M) = p, p(M^c) = 1 - p \). For 2:

\[
p(A|A) = p(A|A, M)p(M) + p(A|A, M^c)p(M^c) \\
= (0)(p) + (1)(1 - p) = 1 - p \\
p(C|A) = p(C|A, M)p(M) + p(C|A, M^c)p(M^c) \\
= (1/3)(p) + (0)(1 - p) = p/3
\]

For 2': Note that the definition of \( M \) changes. Thus \( p(M) = 4p/3, p(M^c) = 1 - 4p/3 \).

\[
p(A|A) = p(A|A, M)p(M) + p(A|A, M^c)p(M^c) \\
= (1/4)(4p/3) + (1)(1 - 4p/3) = 1 - p \\
p(C|A) = p(C|A, M)p(M) + p(C|A, M^c)p(M^c) \\
= (1/4)(4p/3) + (0)(1 - 4p/3) = p/3
\]

Thus the conditional probabilities of the next letter given the previous letter are the same for both 2 and 2'. Hence 2 and 2' are equivalent.

Problem 2

For the analysis, it is easier to use 2' rather than 2.

There are 1 generations from the root to a leaf. If there is even one mutation in any of the generations leading from the root to the leaf, the leaf will be randomly chosen independent of the root. Otherwise, the leaf is equal to the root. Let \( M \) denote the event that there is a mutation in at least one generation from the root to the leaf. Let \( E \) denote the event that the leaf is different from the root.

\[
p(E) = p(E|M)p(M) + p(E|M^c)p(M^c) \\
= (3/4)p(M) + (0)(p(M^c))
\]
We note that $M^c$ occurs if and only if there is no mutation in $l$ generations. Therefore,

$$p(M^c) = (1 - 4p/3)^l$$
$$p(M) = 1 - (1 - 4p/3)^l$$

Thus $p(E) = \frac{3}{4}(1 - (1 - 4p/3)^l)$.

**Problem 3**

If a die is thrown $n$ times, we expect the number of 1’s to be $n/6$. In general, if $X$ is a binomial random variable with parameters $n, p$, then $E[X] = np$.

Just observe that each genome is $n$ letters long, and each letter constitutes an independent bernoulli trial. In the previous problem, we computed the probability that letter $k$ of leaf $i$ is different from letter $k$ of the root. The random variable $X_i$ is a binomial distribution with parameters $n, \frac{3}{4}(1 - (1 - 4p/3)^l)$.

$$E[X_i] = n \left[ \frac{3}{4} \left(1 - (1 - 4p/3)^l\right) \right]$$

**Problem 4**

This problem asks us to simulate the experiment and use it to confirm the answer to the previous problem.

We first write a Matlab function to simulate the experiment.

```matlab
% Simulate the leaves and root given problem parameters of phylogenetic
% tree for HW 4
function [root,leaves]=simulate_tree(n,L,p)
    root=rand(1,n);
    root=(root>0.75)+(root>0.5)+(root>0.25);
    leaves=root;
    for i=1:L
        % Generate 2 children for each parent generation
        leaves=[leaves;leaves];
        % Generate and add noise
        mutate=rand(size(leaves))<p;
        noise=rand(size(leaves)).*mutate;
        noise=(noise>0.75)+(noise>0.5)+(noise>0.25);
        leaves(mutate)=noise(mutate);
    end
end
```

Note that Matlab functions must be stored in separate files and the file name must be the same as the function name. Also the file should be in the same directory as the script with the calling function (or should be added to Matlab’s path).

```matlab
root=rand(1,n);
root=(root>0.75)+(root>0.5)+(root>0.25);
```
randomly generates a root sequence of \( n \) letters: \( 1 \times n \). Each entry is in \( \{0, 1, 2, 3\} \), corresponding to \( \{A, C, T, G\} \) respectively. Then we initialize leaves:

\[
\text{leaves} = \text{root};
\]

The variable ‘leaves’ will finally be a \( 2^l \times n \) matrix with 1 row for every leaf node.

To give birth to a child generation:

\[
\begin{align*}
% & \text{Generate 2 children for each parent generation} \\
\text{leaves} &= [\text{leaves}; \text{leaves}]; \\
% & \text{Generate and add noise} \\
\text{mutate} &= \text{rand(size(leaves))} < p; \\
\text{noise} &= \text{rand(size(leaves))} \cdot \text{mutate}; \\
\text{noise} &= (\text{noise} > 0.75) + (\text{noise} > 0.5) + (\text{noise} > 0.25); \\
\text{leaves(mutate)} &= \text{noise(mutate)};
\end{align*}
\]

The first line simply replicates each leaf twice:

\[
\text{leaves} = [\text{leaves}; \text{leaves}];
\]

Then we decide which letters of the leaf need to be mutated by tossing a biased coin:

\[
\text{mutate} = \text{rand(size(leaves))} < p;
\]

The next two lines are simply generating ‘noise’: A letter chosen at random among \( \{0, 1, 2, 3\} \). Finally we let

\[
\text{leaves(mutate)} = \text{noise(mutate)};
\]

In Matlab, if \( I, M, N \) are matrices with the same size and \( I \) is a matrix with entries 0 or 1, then the assignment \( M(I) = N(I) \) assigns each location in \( M \) where \( I = 1 \) with the corresponding entry from \( N \).

Now we are ready to code the rest of the problem. Just don’t forget that the ‘simulate-tree’ function used Model 2’, hence we must replace \( p \) by \( 4p/3 \). Here is the code:

\[
\begin{align*}
P &= [0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5]; \\
\text{pv} &= 4 \times P / 3; \\
\text{for iter} &= 1: \text{length(pv)} \\
& p = \text{pv(iter);} \\
& [\text{root}, \text{leaves}] = \text{simulate_tree(n,L,p);} \\
& x = \text{zeros(1, size(leaves,1));} \\
& \text{for } i = 1: \text{size(leaves,1)} \\
& \quad x(i) = \text{sum(leaves(i,:)} \sim \text{root}); \\
\end{align*}
\]

\[
\begin{align*}
\text{figure;} \\
\text{hist(x);} \\
\text{ex} &= 3 \times n / 4 \times (1 - (1 - p)^L) \quad \% \text{Note that we have already multiplied by } 4/3 \text{ in the line } \text{pv} = 4 \times P / 3
\end{align*}
\]

You will expect to see the samples \( X_i \) clustered evenly around the mean \( E[X] \).
Here is the plot for $p = 0.1$ and therefore $E[X] = 57.0699$.

Here is the plot for $p = 0.2$ and therefore $E[X] = 71.6265$. 
Here is the plot for $p = 0.3$ and therefore $E[X] = 74.5465$.

Here is the plot for $p = 0.4$ and therefore $E[X] = 74.9633$. 
Here is the plot for $p = 0.5$ and therefore $E[X] = 74.9987$.

Problem 5

In this problem, we try to estimate the root given the leaf. The code is given below:

```matlab
P=0.02:0.02:0.5;
EZ=zeros(size(P));
py=4*P/3;
niter=10;
for i=1:length(py)
    p=py(i);
    sumZ=0;
    for j=1:niter
        [root, leaves]=simulate_tree(n,L,p);
        count=[sum(leaves==0); sum(leaves==1); sum(leaves==2); sum(leaves==3)];
        [dump estimate]=max(count);
        Z=sum((estimate-1)*root)/n;
        sumZ=sumZ+Z;
    end
    EZ(i)=sumZ/niter;
end
plot(P,EZ)
```

The explanation is as follows: ‘leaves’ is a $2^L \times n$ matrix where each row represents a particular leaf. As discussed before, ‘sum()’ will return the column sum of a matrix. Thus the $i$-th entry of ‘sum(leaves==0)’ returns the number of leaves that have 0 or $A$ in their $i$-th letter. ‘max()’
also returns the max in each column. Further ‘[m index]=max(Matrix)’ returns the indices of the maximum (in this case, the row number of the maximum in a particular column). Since the row number of ‘count’ corresponds to (letter+1), (estimate-1) will be the required estimate of the ‘root’ sequence. The rest of the code should be self explanatory.

Now let’s view the plot of $E[Z]$ vs $p$.

As expected, for large $p$, the error is 75%, which is what you expect when the leaves are independent of the root sequence.