1) Solution of practice midterm.

Here are some errors:

a) The "basis case" should be for length \( \ell \leq 2 \), not 3.

b) The "basis case" should assign \( y = x \), and not \( x = y \).

c) \( N' \) should be the ceiling or floor (all integers) of \( N/2 \).

d) The merge should be of \( y_1 \) and \( y_2 \), and not \( x_1 \) and \( x_2 \).

2) To keep things simple, first consider \( X \subseteq \{0, 1\}^n \) with \( N_0 \) and \( N_1 \) being the number of times 0 and 1 appear in \( X \), respectively. Define the empirical parameter as

\[ \hat{\Theta} = \frac{N_1}{N_0 + N_1} \]

where we note that \( N = N_0 + N_1 \).

Our model class is a set or collection of possible representation levels for \( \hat{\Theta} \). We've discussed that a good model class obeys

\[ |class| = O(\sqrt{N}) \]

in which case roughly \( \log_{2}(N) \) bits are needed to encode the appropriate representation level.

But here \( x_{mk} \) depends on the previous \( k \) symbols, and there are \( C_k \) combinations of these \( k \) symbols.
moreover, each conditional distribution provides
probabilities for $k$ possible characters. With
a binary alphabet, i.e., fully, one parameter
captures $n_1$ and $n_2$. With $k$ characters
there are $k!$ parameters.

In total, there are $k!$ parameters for
each of the $C_k$ possible symbols (preceding
$k$ symbols), and the model complexity is
"something like" $\frac{1}{2} C_k \cdot (k-1) \log_2(n)$ bits.

My "something like" alludes to some potential
for reducing the model class. For example,
the set of $k!$ parameters sits on a simplex
because their sum is no more than 1. Therefore,
$\frac{k-1}{2} \log_2(n)$ could probably be reduced to
something like $\frac{k-1}{2} \log_2 (n^k)$. In any case,
such refinements are not covered in our course.

Recall that $N(\mu, \sigma^2)$ can be expressed as
a probability density function (pdf),
$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$ 
Therefore,
$$f_{true}(x) = \frac{1}{\sqrt{\pi} \mu^2} e^{-\frac{x^2}{\mu^2}} + \frac{1}{\sqrt{\pi} \sigma^2} e^{-\frac{(x-\mu)^2}{\sigma^2}}.$$
$$f_{red}(x) = \frac{1}{\sqrt{\pi} \sigma^2} e^{-\frac{x^2}{\sigma^2}}.$$
The probability of red conditioned on observing
X can be computed with Bayes' rule:

\[
\Pr(\text{red} \mid X) = \frac{\Pr(\text{red}, X)}{\Pr(X)}
\]

\[
= \frac{\Pr(\text{red}, X)}{\Pr(\text{red}, X) + \Pr(\text{blue}, X)}
\]

\[
= \frac{\Pr(\text{red}) \cdot \Pr(X \mid \text{red})}{\Pr(\text{red}) \Pr(X \mid \text{red}) + \Pr(\text{blue}) \Pr(X \mid \text{blue})}
\]

we were told that \(\Pr(\text{blue}) = \Pr(\text{red}) = 0.5\)

and so we have

\[
\Pr(\text{red} \mid X) = 0.5 \cdot \frac{\Pr(\text{red}, X)}{0.5 \cdot \Pr(\text{red}, X) + 0.5 \cdot \Pr(\text{blue}, X)}
\]

\[
= 0.5 \left[ \frac{e^{-\alpha x} + e^{-\beta x}}{0.5[e^{-\alpha x} + e^{-\beta x}] + [e^{-\alpha x}]} \right]
\]

we can implement this calculation in MATLAB and see numerically where \(\Pr(\text{red} \mid X)\) is less than or greater than 0.5. When it exceeds 0.5, a "red" guess is reasonable, else blue is the more likely posterior.

The algorithm has two steps:
step 1: run merge sort in \(\Theta(\log n)\) time.
step 2: we have a list of \(N\) sorted numbers, and the numbers being distinct or not can be decided by scanning through the \(n-1\) adjacent pairs and checking whether their values are identical.
Here is possible pseudo-code:

```plaintext
for index = 1 to n-1
    if X(index) = X(index+1)
        then there is a repeat occurrence
            break / return
    else continue looping
```

It can be shown that \( T(n + n \log(n)) = \Theta(n \log(n)) \)

To see why, note that

\[
 n \log_2(n) \leq n \log_2(e) \leq 2n \log_2(n)
\]

\[
\uparrow \quad \uparrow
\]

for \( n \geq 2 \quad \text{for} \quad n \\leq 2
\]

Finally, let's show why we write \( \Theta(n \log(n)) \)
without specifying the base of the logarithm.

To see why, we will prove that \( \log_4(n) = \Theta(\log_2(n)) \).

Recall that \( \log_4(n) = \frac{\log_2(n)}{\log_2(4)} = \frac{\log_2(n)}{2} \).

Therefore, we can take \( N_0 = 1 \) and \( c_1 = \frac{1}{2} \),
we have that 
\[
\frac{1}{2} \log_2(n) \leq \log_4(n) = \frac{1}{2} \log_2(n) \leq \frac{1}{2} \log_2(n)
\]

for all \( n \geq N_0 \). This proves formally that
\( \log_4(n) = \Theta(\log_2(n)) \), and the result can be expanded to any logarithmic base.
we will show formally that
\[ 0.001 n^3 + 100 n^2 = \Theta(n^3). \]
we want to identify some \( N_0 \geq 0, c_1 \geq 0, \) and \( c_2 \geq 0 \) such that
\[ c_1 n^3 \leq 0.001 n^3 + 100 n^2 \leq c_2 n^3 \]
for all \( n \geq N_0. \) Let's take \( c_1 = 0.001, \) and we can see that
\[ c_1 n^3 = 0.001 n^3 \leq 100 n^2 + 0.001 n^3 \]
for all \( n > 0. \)

What about \( c_2? \) How about \( c_2 = 0.000 \) which is the sum of the two coefficients in the polynomial?
\[ 0.001 n^3 + 100 n^2 = 0.001 n^3 + 100 n^2 \]
\[ n=1 \] note \( n^3 \) instead of \( n^2 \)
\[ = 100.001 n^3 \]
\[ = c_2 n^3. \]
Therefore, in summary, we have shown for \( N_0 = \max(0,1) = 1, c_1 = 0.001, \) and \( c_2 = 100.001 \)
that
\[ c_1 n^3 = 0.001 n^3 \leq 100 n^2 + 0.001 n^3 \leq 100.001 n^3 \]
\[ = c_2 n^3 \]
for all \( n \geq N_0. \) This completes the formal demonstration that \( 0.001 n^3 + 100 n^2 = \Theta(n^3). \)