1. **Mutation and Duplications** (*Collaboration is not allowed on this Problem*)

Motivated by DNA mutations, we consider strings of symbols and study point mutations and tandem duplication processes. A point mutation is a change in a single symbol of a given string. For example, for the binary string 0011101, a point mutation in the fourth location results in 0010101. A tandem duplication of length \( k \), is a duplication of a substring of length \( k \) next to its original position. For example, a tandem duplication of length 3 starting in the second location of 0011101 results in 0011011101.

In class we proved that every binary string can be generated by tandem duplication steps from one of the following six strings \{0, 1, 01, 10, 010, 101\} (that we call seeds). The generation process from the seeds is not unique. For example, consider the string 001001. It can be generated from 01 in two ways:

\[
01 \rightarrow 0101 \rightarrow 001001
\]

requiring 3 tandem duplication steps, and

\[
01 \rightarrow 001 \rightarrow 001001
\]

requiring 2 tandem duplication steps.

The duplication distance is defined as the minimum number of steps required to generate a given string from a seed by a tandem duplication process. In the example above, the duplication distance of 001001 is 2.

(a) Find the duplication distance of the following strings. Show the steps leading from the seed to the string.

i. 00010101
ii. 0001100001
iii. 0001010111
iv. 0100000101011
v. 0010001100100111
vi. 0110100110010110

(b) Here we allow point mutations and define the mutation distance as the minimum number of the sum of tandem duplications steps and point mutations steps required to generate a string from a given seed. For example, the duplication distance of 111 is 2 (from the seed 1), while the mutation distance is 1 (from the seed 101).

Find the mutation distance of the following strings from the seed 01. Show the steps leading from the seed to the string.

i. 0100111
ii. 00010101
iii. 0001100001
iv. 0001010111
v. 1010111000

(c) A duplication of a substring of length \( k \) next to its original position creates a tandem repeat. Here we consider the reverse process. Namely, given a string with tandem repeats, in each step we remove one copy of a repeat (in some order) until the string does not have any tandem repeats. We call a string with no tandem repeats a squarefree string. For example:

\[
010000101 \rightarrow 010101 \rightarrow 0101 \rightarrow 01
\]

The removal of a copy of a tandem repeat is called deduplication and the final squarefree string obtained is the seed of \( x \). Hence, the seed of 01000101 is 01. In class we proved that every binary string has a unique seed and it belongs to the set \{0, 1, 01, 10, 010, 101\}. However, for strings over ternary (or higher) alphabet, there can be multiple seeds. For example the ternary string \( y = 012101212 \), has two seeds:

\[
012101212 \rightarrow 0121012 = Seed 1
\]

\[
012101212 \rightarrow 0121012 = Seed 2
\]

Find all possible deduplication seed(s) of the following strings. Show your work.

i. 30121013012101212
ii. 123032123032303
iii. 120312031230321230323031212

(d) In this question, we consider quaternary alphabet (an alphabet of size 4, like the DNA) and study the closest ancestor of two strings (species). Specifically, we consider the mutation distance between two strings \( x \) and \( z \) as the minimum number of point mutations and tandem duplications required to generate \( x \) from \( z \), we denote it by \( d_m(z, x) \). For example, let \( x = 12301233 \) and \( z = 1230 \), then \( d_m(z, x) = 2 \):

\[
1230 \rightarrow 12301230 \rightarrow_{\text{pmut}} 12301233
\]
Consider two strings $x_1$ and $x_2$, the closest ancestor of $x_1$ and $x_2$ is the string $y$, other than $x_1$ and $x_2$, which minimizes $d_m(y, x_1) + d_m(y, x_2)$, namely,

$$y = \arg\min_{z \neq x_1, z \neq x_2} [d_m(z, x_1) + d_m(z, x_2)]$$

For example, let $x_1 = 1233$ and $x_2 = 12301233$. The closest ancestor of $x_1$ and $x_2$ is $y = 1230$, with $d_m(y, x_1) = 1$ and $d_m(y, x_2) = 2$

Find the closest ancestor of the following strings. Show your work.

i. $123012301203$ and $123012031202$
ii. $1230123212331232$ and $1230123312301233$

2. The Eraser-Monkey  
(Collaboration is allowed on this problem)

The world is not perfect, specifically; errors happen. In this problem you will learn about simple schemes for error correction. A useful operation for error correction is the parity function on binary variables (bits), also known as the eXclusive OR (XOR) function, defined as follows:

Consider the $n$ binary variables, $x_i \in \{0, 1\}$, $1 \leq i \leq n$, let $X = (x_1, x_2, \ldots, x_n)$ then

$$XOR(X) = \begin{cases} 1 & \text{if the number of 1's in } X \text{ is odd} \\ 0 & \text{otherwise} \end{cases}$$

We use the following notation, $XOR(x_1, x_2) = x_1 \oplus x_2$. For example, $1 \oplus 1 \oplus 0 = 0$ and $0 \oplus 1 \oplus 0 = 1$.

Assume that as part of your class project you have written the following table.

$$A = \begin{array}{cccc}
0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 \\
\end{array}$$

While you were away from your room, your roommate who likes to play with erasers (the so called eraser-monkey), has erased one of the columns in the table. When you come back to your room you are mortified to discover $\hat{A}$. The $Es$ represent the blank (erased) space.

$$\hat{A} = \begin{array}{cccc}
0 & E & 1 & 1 \\
0 & E & 1 & 0 \\
1 & E & 1 & 1 \\
1 & E & 1 & 1 \\
\end{array}$$

To battle the eraser-monkey you decide to add a fifth column (called a parity column) that contains the row parity, as follows:

$$\begin{array}{cccc}
a_1 & b_1 & c_1 & d_1 \\
a_2 & b_2 & c_2 & d_2 \\
a_3 & b_3 & c_3 & d_3 \\
a_4 & b_4 & c_4 & d_4 \\
\end{array} \quad \begin{array}{c}
a_1 \oplus b_1 \oplus c_1 \oplus d_1 \\
a_2 \oplus b_2 \oplus c_2 \oplus d_2 \\
a_3 \oplus b_3 \oplus c_3 \oplus d_3 \\
a_4 \oplus b_4 \oplus c_4 \oplus d_4 \\
\end{array}$$
**Important note:** The eraser-monkey can erase entries in any column - including a parity column. This note holds for all the parts of this problem.

(a) Show the matrix $A$ with the additional parity column. Explain how to recover the missing bits after the eraser-monkey erases the second column. Can you correct any column erasure? Why?

(b) Assume that you have an $n \times n$ table with an additional parity column. We define an erasure pattern as a subset of entries in the $n \times (n + 1)$ table. The number of entries in an erasure pattern, denoted by $p$, can be arbitrary, namely $1 \leq p \leq n^2 + n$. In (a) you considered an erasure pattern that is confined to a single column. Here you are asked to classify all the erasure patterns that can be corrected in an $n \times n$ table with an additional parity column. What are the correctable erasure patterns? How many different correctable erasure patterns are there?

(c) Assume that instead of writing bits in the table you need to write single decimal digits, namely $\{0, 1, \ldots, 9\}$. How will you protect your data from a single column erasure by adding a single column? Please note that the additional column must contain only single decimal digits. Justify your solution.

(d) The eraser-monkey discovered that you are able to recover any single column erasure and is very upset. To calm himself down he decides to erase two adjacent columns. Propose a method (by adding two additional columns) for correcting two-adjacent-columns erasures for an arbitrary $n \times n$ table. Assume that the entries in the table are binary. Prove that your method works.

(e) Consider the following $2 \times 4$ table. Assume that the entries in the table are binary. Notice that there are two information columns and two parity columns. Can it correct any two-columns erasure (not necessarily adjacent)? If your answer is positive, provide a proof. If your answer is negative, provide an example of an uncorrectable two-columns erasure.

\[
\begin{array}{cccc}
a & c & a \oplus c & b \oplus c \\
b & d & b \oplus d & a \oplus d \\
\end{array}
\]

(f) Consider the following $2 \times 4$ table. Assume that the entries in the table are binary. Notice that there are two information columns and two parity columns. Can it correct any two-columns erasure (not necessarily adjacent)? If your answer is positive, provide a proof. If your answer is negative, provide an example of an uncorrectable two-columns erasure.

\[
\begin{array}{cccc}
a & c & a \oplus c & b \oplus c \\
b & d & b \oplus d & a \oplus b \oplus d \\
\end{array}
\]