Solution Guide: Parity Error Detection and Correction Exercise

This document provides the solution and theoretical explanations for the `parity_exercise.py`.

`parity_solution.py`

This file contains the complete, commented implementation for the exercise. Each `TODO` section has been filled in with the correct logic. The comments aim to explain *why* the code works, especially for the core concepts of odd parity and error correction.

How to Run

The solution script is self-contained. To run the full demonstration and see the output, simply execute it from your terminal:

```
python3 parity_solution.py
```

The script will:

1. Demonstrate simple column parity with both single-bit and burst errors.
2. Demonstrate 2D parity, showing its ability to correct a single-bit error.
3. Print the theoretical questions for you to answer based on the output.

Answers to Theoretical Questions

Here are the explanations for the questions posed at the end of the script.

1. How does simple column parity behave with single bit errors?

It **always detects** a single bit error. A single flipped bit will change the sum of its column by one, flipping the sum from even to odd or vice-versa. This will cause the recalculated parity bit for that column to be different from the one that was received, reliably signaling an error.

2. How does simple column parity behave with burst-like errors?

Its behavior depends on how the burst affects the columns.

- **Generally Detected:** Most burst errors **will be detected**. A burst of 3 consecutive bits, for example, will flip one bit in three different columns (if it stays on one row). This changes the parity of all three columns, and the error is easily detected.
- **Potentially Undetected:** An error can go **undetected** only if an **even number of bits are flipped in *every single column***. For a burst error, this is a very specific and rare pattern. For example, a burst of 4 that flips two bits in column 0 and two bits in column 1 would be missed, but a consecutive burst is unlikely to do this. A burst of length 8 that flips exactly two bits in every column would also be missed.

3. What is the maximal length of a bit sequence for which we can guarantee the detection of errors?

The maximal length of a burst error for which we can **guarantee detection** is **L (the number of columns)**, which is 4 in our example.

- **Why?** A consecutive burst of length `L` or less that occurs on a single row will flip exactly one bit in `L` or fewer columns. Since no column has more than one bit flipped, the parity check is guaranteed to fail for each affected column, detecting the error.
- The guarantee is lost if the burst is long enough to potentially cause an even number of errors in one or more columns (e.g., a burst of length `L+1` could wrap to the next row and land in the same column as its first bit).

4. How can the 2D parity technique fix a single bit error?

2D parity uses a grid system to pinpoint the exact location of a single error. The logic is as follows:

1. **Find the Bad Row:** The system recalculates the parity for each row. The one row where the calculated parity does not match the received parity bit is the row containing the error.
2. **Find the Bad Column:** It does the same for each column. The one column where the parity is incorrect is the column containing the error.
3. **The Intersection:** The single bit that is at the **intersection** of the bad row and the bad column is the faulty bit. By flipping this one bit (0 to 1 or 1 to 0), the error is corrected.

5. Can we use 2D parity in the case of multiple faulty bits and burst-like errors?

No, 2D parity **cannot reliably fix** multiple errors.

- **Detection:** It can often *detect* that multiple errors have occurred. For example, if two bits are flipped, you might find that two rows and two columns have incorrect parity checks.
- **Correction:** However, it cannot *correct* them because there is no single intersection point. With two bad rows and two bad columns, there are four possible locations for the errors, and the system has no way of knowing which two are the correct ones to flip. A burst error is just a specific type of multiple-bit error, so it also cannot be corrected.