# Experiment 1: Error Detection using Parity

**Aim:** To apply Parity check rules for error detection

**Objective:** After carrying out this experiment, students will be able to:

* Apply 1D and 2D parity rules for error detection
* Analyze the difference between 1D and 2D parity and their limitations

**Problem statement:** You are required to write separate programs to demonstrate the use of 1D and 2D parity. Take the input bit streams (max five) of 7 bits each from the user. Your programs should calculate the parity and display the input and output bit streams.

**Analysis:** While analyzing your program, you are required to address the following points:

* Why can this method not be used to correct errors?
* How are 1D and 2D parity different?
* What are the limitations of this method of error detection?

**MARKS DISTRIBUTION**

|  |  |  |
| --- | --- | --- |
| **Component** | **Maximum Marks** | **Marks Obtained** |
| Algorithm/Flowchart | 7 |  |
| Program | 7 |  |
| Results/ Documentation | 6 |  |
| **Total** | **20** |  |

Submitted by: Harshit Kumar

Register No: 21ETMC412011

# Algorithm/Flowchart

**Step-01:-(1D Parity)**

 At sender side,

* Total number of 1’s in the data unit to be transmitted is counted.
* The total number of 1’s in the data unit is made even in case of even parity.
* The total number of 1’s in the data unit is made odd in case of odd parity.
* This is done by adding an extra bit called as **parity bit**.

**Step-02:**

* The newly formed code word (Original data + parity bit) is transmitted to the receiver.

**Step-03:**

At receiver side,

* Receiver receives the transmitted code word.
* The total number of 1’s in the received code word is counted.

Then, following cases are possible-

* If total number of 1’s is even and even parity is used, then receiver assumes that no error occurred.
* If total number of 1’s is even and odd parity is used, then receiver assumes that error occurred.
* If total number of 1’s is odd and odd parity is used, then receiver assumes that no error occurred.
* If total number of 1’s is odd and even parity is used, then receiver assumes that error occurred.

**2D Parity:-**

**(Step-1)input:**

* Accept the number of data segments, let's call it num\_segments.
* Create an empty matrix mat of size num\_segments x segment\_length where segment\_length is the length of each data segment.

**(Step-2)Populate Matrix with Data Segments:**

* For each data segment, repeat the following steps:
* Accept the segment input. Ensure all segments are of the same length.
* Calculate and append the even parity bit for the segment.
* Add the segment with the parity bit to the matrix.

**(Step-3)Calculate Horizontal Parity:**

* For each column in the matrix, calculate the even parity bit by summing up the bits in that column.
* Append the horizontal parity bit to the matrix as a new row.

**(Step-4)Input New Data Segments:**

* Create another matrix mat1 for new data segments using the same procedure as in step 2.

**(Step-5)Calculate Vertical Parity:**

* For each row in mat1, calculate the even parity bit by summing up the bits in that row.
* Append the vertical parity bit to mat1 as a new column.

**(Step-6)Check for Errors:**

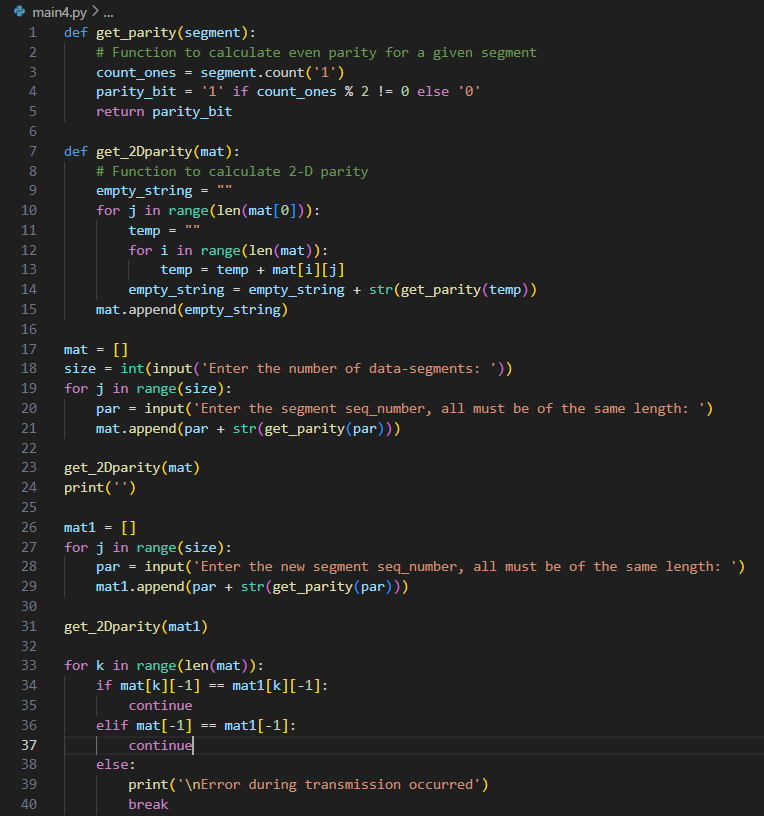
* Compare the horizontal parity bits of the original matrix mat and the calculated horizontal parity bits of mat1.
* If they match, continue to the next step.
* If not, print an error message and exit.
* Compare the vertical parity bits of the original matrix mat and the calculated vertical parity bits of mat1.
* If they match, transmission is successful.
* If not, print an error message indicating an error during transmission.

# Program

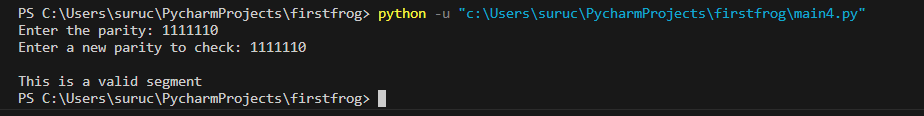
1-D parity



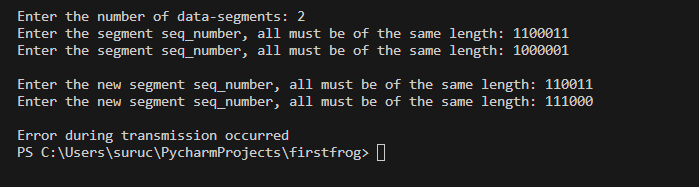
2-D Parity



# Results

Output for 1D Parity:-

Output for 2D Parity:-



# Analysis and Discussions

Analysis for 1st one

If two bits are transposed (change places) then the computer could be fooled into thinking the data is correct and not corrupted.

If two random bits change state then the system could also be fooled.

Analysis for 2nd one

Single Parity Check

* Single Parity checking is the simple mechanism and inexpensive to detect the errors.
* In this technique, a redundant bit is also known as a parity bit which is appended at the end of the data unit so that the number of 1s becomes even. Therefore, the total number of transmitted bits would be 9 bits.
* If the number of 1s bits is odd, then parity bit 1 is appended and if the number of 1s bits is even, then parity bit 0 is appended at the end of the data unit.
* At the receiving end, the parity bit is calculated from the received data bits and compared with the received parity bit.
* This technique generates the total number of 1s even, so it is known as even-parity checking.

Two-Dimensional Parity Check

* Performance can be improved by using **Two-Dimensional Parity Check** which organizes the data in the form of a table.
* Parity check bits are computed for each row, which is equivalent to the single-parity check.
* In Two-Dimensional Parity check, a block of bits is divided into rows, and the redundant row of bits is added to the whole block.
* At the receiving end, the parity bits are compared with the parity bits computed from the received data.

Analysis for 3rd one

Drawbacks Of Single Parity Checking

* It can only detect single-bit errors which are very rare.
* If two bits are interchanged, then it cannot detect the errors

Drawbacks Of 2D Parity Check

* If two bits in one data unit are corrupted and two bits exactly the same position in another data unit are also corrupted, then 2D Parity checker will not be able to detect the error.
* This technique cannot be used to detect the 4-bit errors or more in some cases.

# Conclusions

From the above discussion and analysis we were able to

* Apply 1D and 2D parity rules for error detection
* Analyse the difference between 1D and 2D parity and their limitations